Application Note 227

Using the Microcontroller Prototyping System with the example reference design

Document number: ARM DAI0227A
Issued: August 2009
Copyright ARM Limited 2009

ARM

Application Note 227
Using the Microcontroller Prototyping System with the example reference design

Copyright © 2009 ARM Limited. All rights reserved.

Release information

The following changes have been made to this Application Note.

Change history

Date	Issue	Change
August 2009	A	First release

Version controlled by Domino.Doc DS158-GENC-009799 0.2

References

	Document	Issuer
[1]	User Manual for HMALC-AS3-52	Gleichmann Industries
[2]	User Manual for Hpe-midi-V2	Gleichmann Industries
[3]	HPE_Desk-Basic Online Help	Gleichmann Industries
[4]	Altera Double Data Rate I/O Megafunctions User Guide	Altera Corporation
[5]	The Definitive Guide to the ARM Cortex-M3 ISBN: 978-0-7506-8534-4	Elsevier (by Joseph Yiu)
[6]	PrimeCell® Synchronous Serial Port (PL022) Technical Reference Manual	ARM Ltd.
[7]	CH7303 HDTV / DVI Transmitter (CH7303) Data Sheet	Chrontel
[8]	ARM Dual-Timer Module (SP804) Technical Reference Manual	ARM Ltd.
[9]	PrimeCell® Real Time Clock (PL031) Technical Reference Manual	ARM Ltd.
[10]	ARM Watchdog Module (SP805) Technical Reference Manual	ARM Ltd.
[11]	PrimeCell® UART (PL011) Technical Reference Manual (Revision: r1p5)	ARM Ltd.
[12]	PrimeCell® Advanced Audio CODEC Interface (PL041) Technical Reference Manual	ARM Ltd.
[13]	ARM PrimeCell Multimedia Card Interface (PL181) Technical Reference Manual	ARM Ltd.
[14]	ISP1761 Hi-Speed Universal Serial Bus On-The-Go controller. Rev. 05 — 13 March 2008 Product data sheet.	ST-NXP Wireless

Proprietary notice

ARM, ARM Powered, StrongARM, Thumb, Multi-ICE, ModelGen, PrimeCell, PrimeXsys, RealView, TrustZone, Jazelle, ARM7TDMI, ARM9TDMI, ARMulator AMBA, and The Architecture for the Digital World are registered trademarks of ARM Limited. Cortex, AXI, AHB, ARM7, ARM7TDMI-S, ARM7EJ-S, ARM720T, ARM740T, ARM9, ARM9TDMI, ARM920T, ARM922T, ARM940T, ARM9E, ARM9E-S, ARM926EJ-S, ARM946E-S, ARM966E-S, ARM968E-S, ARM996HS, ARM10, ARM1020E, ARM1022E, ARM1026EJ-S, ARM11, ARM1136J-S, ARM1136JF-S, ARM1156T2-S, ARM1156T2F-S, ARM1176JZ-S, ARM1176JZF-S, EmbeddedICE, EmbeddedICE-RT, AMBA, ARM Development Suite, ETM, ETM7, ETM9, ETM10, ETM10RV, ETM11, Embedded Trace Macrocell, Embedded Trace Buffer, ETB, ETB11, Embedded Trace Kit, Integrator, JTEK, Mali, MultiTrace, MPCore, MOVE, OptimoDE, AudioDE, SecurCore, SC100, SC110, SC200, SC210, Keil, µVision, ULINK are trademarks of ARM Limited. Java is a trademark of Sun Microsystems, Inc. XScale is a trademark of Intel Corporation. All other brand names or product names are the property of their respective holders. “ARM” is used to represent ARM Holdings plc (LSE: ARM and NASDAQ: ARMHY); its operating company ARM Limited; and the regional subsidiaries: ARM, Inc.; ARM KK; ARM Korea Ltd.; ARM Taiwan Limited; ARM France SAS; ARM Consulting (Shanghai) Co. Ltd.; ARM Belgium N.V.; AXYS Design Automation Inc.; ARM Germany GmbH; ARM Embedded Technologies Pvt. Ltd.; ARM Norway, AS; and ARMSweden AB. Neither the whole nor any part of the information contained in, or the product described in, this document may be adapted or reproduced in any material form except with the prior written permission of the copyright holder. The product described in this document is subject to continuous developments and improvements. All particulars of the product and its use contained in this document are given by ARM in good faith. All warranties implied or expressed, including but not limited to implied warranties of satisfactory quality or fitness for purpose are excluded. This document is intended only to provide information to the reader about the product. To the extent permitted by local laws ARM shall not be liable for any loss or damage arising from the use of any information in this document or any error or omission in such information.

Confidentiality status

This document is Open Access. This document has no restriction on distribution.

Feedback on this Application Note

If you have any comments on this Application Note, please send email to errata@arm.com giving:

- the document title
- the document number
- the page number(s) to which your comments refer
- an explanation of your comments.

General suggestions for additions and improvements are also welcome.

ARM web address

http://www.arm.com

Copyright © 2009 ARM Limited. All rights reserved.

Table of Contents

1 INTRODUCTION 1

1.1 Purpose of this application note 1

1.2 Overview of the hardware platform 1

2 HARDWARE DESCRIPTION 5

2.1 Block Diagram 5

2.2 Bus Architecture of DUT FPGA 6

2.3 Clocks and Resets 7

3 PROGRAMMER'S MODEL 10

3.1 Interrupts 10

3.2 Memory map 11

3.3 DUT FPGA Registers 11

4 FPGA DESIGN 18

4.1 Directory structure 18

4.2 Re-building the DUT FPGA 20

4.3 Clock and reset settings 25

5 EXAMPLE SOFTWARE 27

5.1 BootMonitor 27

5.2 Selftest 33

6 EXTERNAL INTERFACES 37

6.1 Clocks and Resets 37

6.2 Processor Interface 37

6.3 Video Interface 38

6.4 Human Interface 39

6.5 FPGA Configuration connections 41

6.6 SEMULATOR connections 41

6.7 USB Interface 41

Copyright © 2009 ARM Limited. All rights reserved.

6.8 Multi-Media/SD Card Interface 42
6.9 Audio AC97 Interface 43
6.10 A/D & D/A Interface 43
6.11 Memory DDR Interface 43
6.12 Ethernet Phy Interface 44
6.13 CAN Interface 44
6.14 Flexray Interface 44
6.15 LIN Interface 45
6.16 RS232 connections 45

Table of Figures

Figure 1: Architecture 2
Figure 2: Microcontroller Prototyping System Front Panel 3
Figure 3: Microcontroller Prototyping System Rear Panel 3
Figure 4: Microcontroller Prototyping System Inside 4
Figure 5: Block diagram of the ARM Microcontroller Prototyping System. 5
Figure 6: Bus Architecture of DUT FPGA 6
Figure 7: MPS Clock and Reset Architecture 7
Figure 8: Hpe_desk clock factory configuration 8
Figure 9: Interrupt Allocation Table 10
Figure 10: DUT FPGA memory map 11
Figure 11: 7 Segment Display Segment Identification 13
Figure 12: Timing Diagram of the I2Cbus 14
Figure 13: Top level directory structure 18
Figure 14: peripherals directory structure 19
Figure 15: fpgu dut directory structure 20
Figure 16 Top level directory structure 27
Figure 17: BootMonitor directory structure 33
Figure 18: Selftest directory structure 35
Figure 19: Video DDR interface 39
Figure 20: HUMI logic 40
Figure 21: MCI interface connections 42
Figure 22: I2C Connections 43

Table of Tables

Table 1: Clock Routing 9
Table 2: Reset Routing 9
Table 3: dut fpga verilog files 20
Table 4: System Configuration 21
Table 5: FPGA scripts 21
Table 6: Clock and Reset Destinations 26
Table 7: Bootmonitor switch utilisation 28
Table 8: Main Menu 29
Table 9: Configure Submenu 30
Table 10: Debug Submenu 31
Table 11: Flash Submenu 32
Table 12: SDCard Submenu 32
Table 13: AACI loopback cable 36
Table 13: UART loopback cable 36
Table 15: Clocks and Resets 37
Table 16: Processor interface 37

Copyright © 2009 ARM Limited. All rights reserved.

Table 17: DUT to CPU FPGA video signals **38**
Table 18: Video and LCD Connections **38**
Table 19: Human Interface **39**
Table 20: FPGA configuration Connections **41**
Table 21: Semulator Connections **41**
Table 22: USB Connections **42**
Table 23: SDCard Connections **42**
Table 24: AC97 Connections **43**
Table 25: A/D & D/A Connections **43**
Table 26: Memory/Childboard Connections **44**
Table 27: Ethernet Phy Connections **44**
Table 28: CAN Connections **44**
Table 29: Flexray Connections **45**
Table 30: LIN Connections **45**
Table 31: RS232 Connections **45**

Copyright © 2009 ARM Limited. All rights reserved.

1 Introduction

The Keil **Microcontroller Prototyping System** (MPS) enables evaluation and prototyping of ARM® Cortex™-M class processors and user defined peripherals in a single product. The MPS is the first prototyping system incorporating a full-speed Cortex-M class processor implemented in FPGA which can be integrated with third-party peripheral IP to deliver a prototyping system for hardware and software application development.

The MPS enables the implementation a Cortex-M class system without needing to have access to the processor RTL, meaning different processors can be benchmarked in order to choose the most suitable for the intended devices price/performance. Additionally, the MPS is delivered fully configured with the Cortex-M processor and is fully tested so that the user does not have to test the processor implementation and can immediately begin adding third-party IP or writing software.

1.1 Purpose of this application note

The application note explains the example DUT FPGA design, how to rebuild it and program it into the MPS.

It will guide you through and explain the;

1. Architecture of the system
    - How the components are interconnected
    - Clock and reset structure
2. Programmers model
    - Interrupt assignment
    - Memory map
    - Peripheral register descriptions
3. FPGA design
    - structure and directory format
    - Rebuilding the FPGA image
    - Configuring Hpe®_desk and MPS
    - Programming the MPS with the new image
4. Example Software
    - BootMonitor
    - Selftest
5. Modifying the design
    - Connecting to the outside world
    - FPGA pin signal assignments

1.2 Overview of the hardware platform

The MPS was developed with two FPGAs to give the maximum flexibility for user prototyping while maintaining IP protection of a Cortex-M class processor (to allow wide customer accessibility without the need to license the processor). The use of the two FPGAs allows the microcontroller, debug and memory subsystem to be implemented in

Copyright © 2009 ARM Limited. All rights reserved.

an encrypted and secure FPGA (CPU) and the user design with access to the peripheral interfaces to be implemented in the non-encrypted FPGA (DUT).

An AHB-Lite interface from the CPU FPGA to the DUT FPGA, together with interrupts and other sideband signals allows the DUT FPGA to interact with and be controlled by the microcontroller as if it were a single System on Chip.

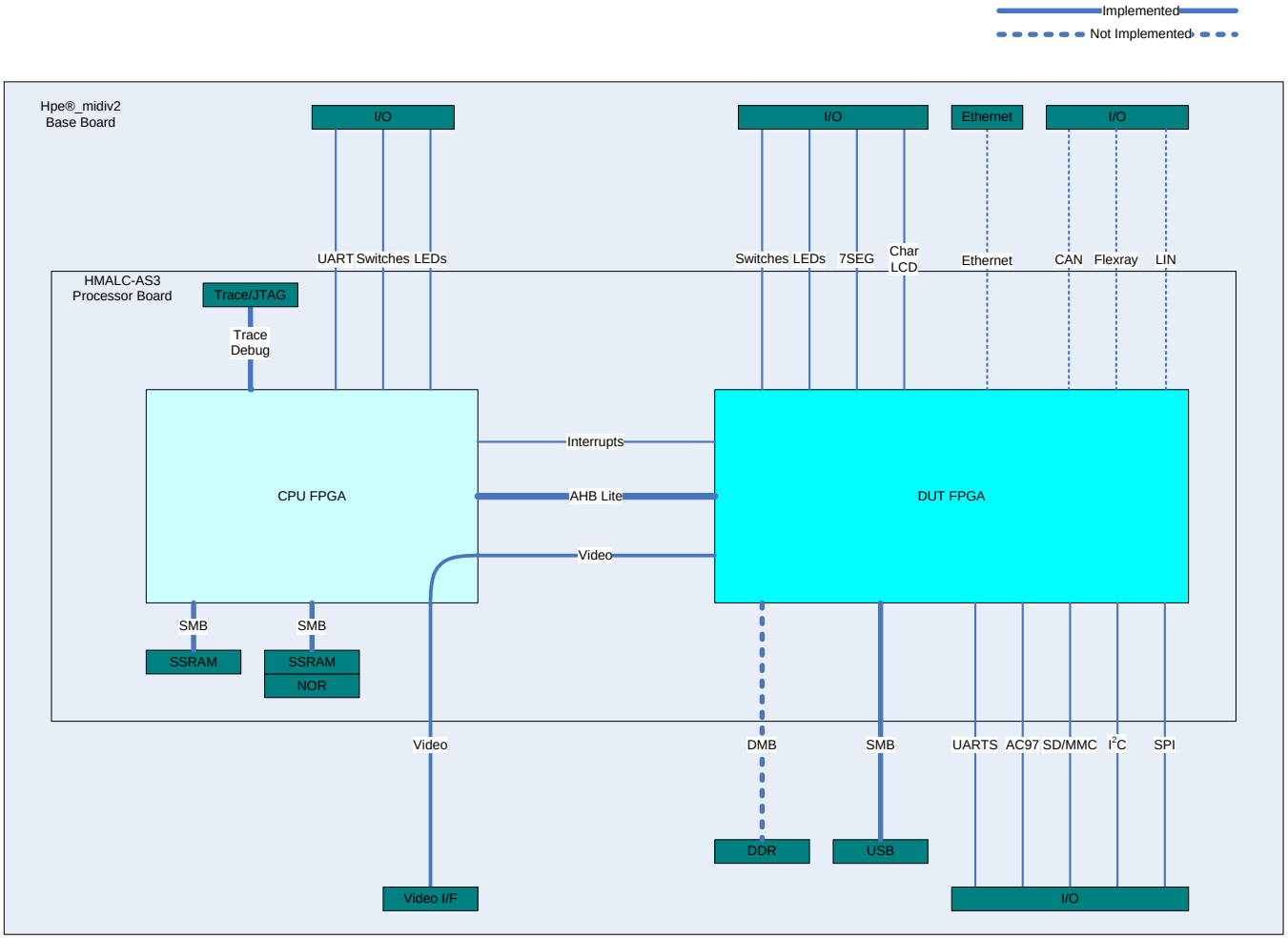


Figure 1: Architecture

The MPS has a front panel (as shown in Figure 2) fitted with;

- Reset button
- LCD character display 2 rows by 40 characters
- 4 press buttons to the DUT FPGA
- 8 LEDs from the CPU FPGA
- 8 LEDs from the DUT FPGA
- LEDs to show if the FPGAs are over temperature
- LEDs to show status of the power supplies
- LEDs to show FPGAs configuration status
- USB OTG connector for configuration/programming

Copyright © 2009 ARM Limited. All rights reserved.

Figure 2: Microcontroller Prototyping System Front Panel

The rear panel (as shown in Figure 3) has the following features;

- Power connector and switch
- USB Host and OTG ports
- SDCard slot
- RS232/CAN/LIN/FlexRay connections
- 10/100 Ethernet RJ45 interface
- RGB video output port (DVI-A connector)
- RS232 ports
- Childboard expansion slot (red plastic plates)
- Audio line in and line out
- JTAG/SWD connector for ULINK-2 and other ARM debug hardware
- External clock source (not supported at present)
- Analogue port (not supported at present)

Figure 3: Microcontroller Prototyping System Rear Panel

Copyright © 2009 ARM Limited. All rights reserved.

The inside of the MPS (as shown in Figure 3) shows the;

- Switches, buttons and 7 segment displays
- Location of the FPGAs (CPU and DUT)
- IDC ribbon cables for selection of RS232/CAN/LIN/Flexray
- Childboard connector for user expansion
- JTAG and Trace (Mictor) connection for debug

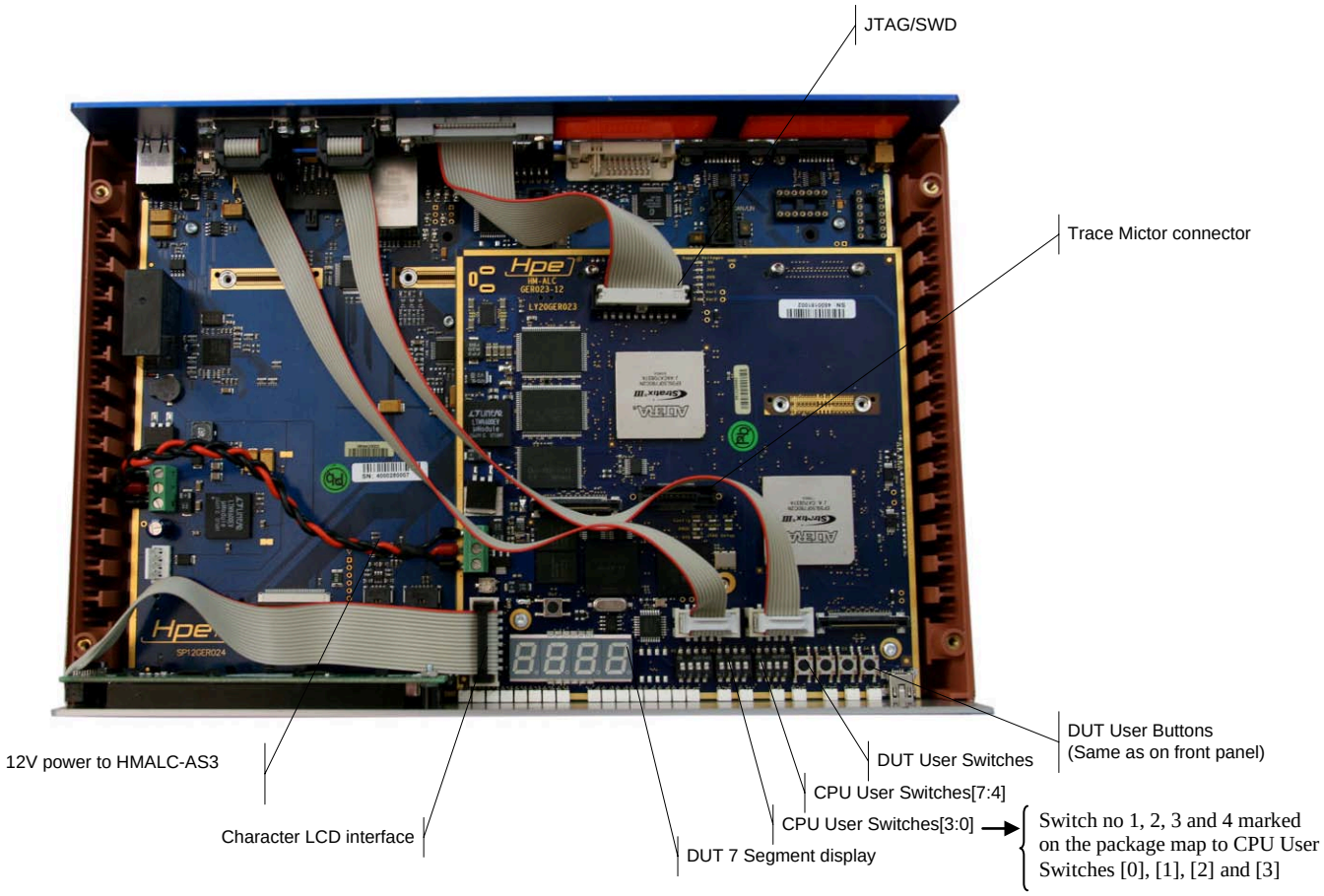

JTAG/SWD
Trace Mictor connector
DUT User Buttons
(Same as on front panel)
DUT User Switches
CPU User Switches[7:4]
CPU User Switches[3:0]
Switch no 1, 2, 3 and 4 marked on the package map to CPU User Switches [0], [1], [2] and [3]
DUT 7 Segment display
Character LCD interface
12V power to HMALC-AS3

Figure 4: Microcontroller Prototyping System Inside

Copyright © 2009 ARM Limited. All rights reserved.

2 Hardware Description

This application note explains the example AHB (AMBA 2.0) system implemented on the MPS. This board contains two FPGAs:

- The CPU FPGA: An ARM Cortex-M class processor with debug and trace support depending on the processor, two memory controllers to operate interfaces to the noBLRAM and FLASH NOR RAM on the board, Touchscreen (SPI) and Video configuration (I^2C) peripherals and a configuration register block. This design is processor specific and more details can be found in the relevant application note.
- The DUT FPGA: Containing an example system including timers, display drivers, an audio interface and an MCI/SD card interface.

2.1 Block Diagram

Figure 4 shows how the two FPGAs are interconnected with a 32bit AHB-Lite interface between the CPU and the DUT FPGAs, also sideband signals and interrupts between the FPGAs dependant on the processor implemented to enable a realistic system in the DUT FPGA.

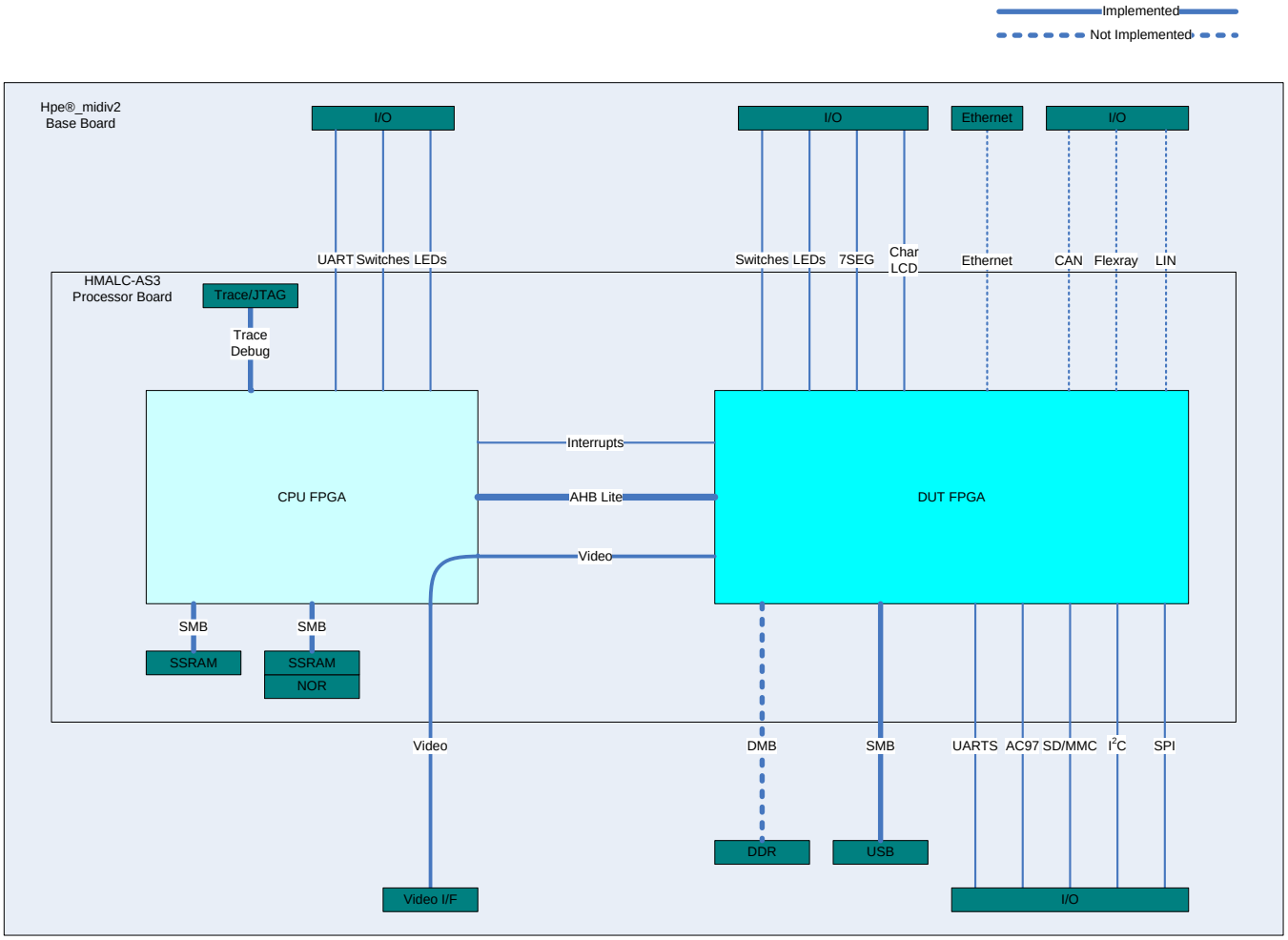


Figure 5: Block diagram of the ARM Microcontroller Prototyping System

Copyright © 2009 ARM Limited. All rights reserved.

Bus architecture

2.2 Bus Architecture of DUT FPGA

An AHB-Lite system is implemented in the DUT FPGA to give the processor access to all the AHB peripherals within the FPGA. An AHB to APB bridge allows implementation of APB peripherals in the system also.

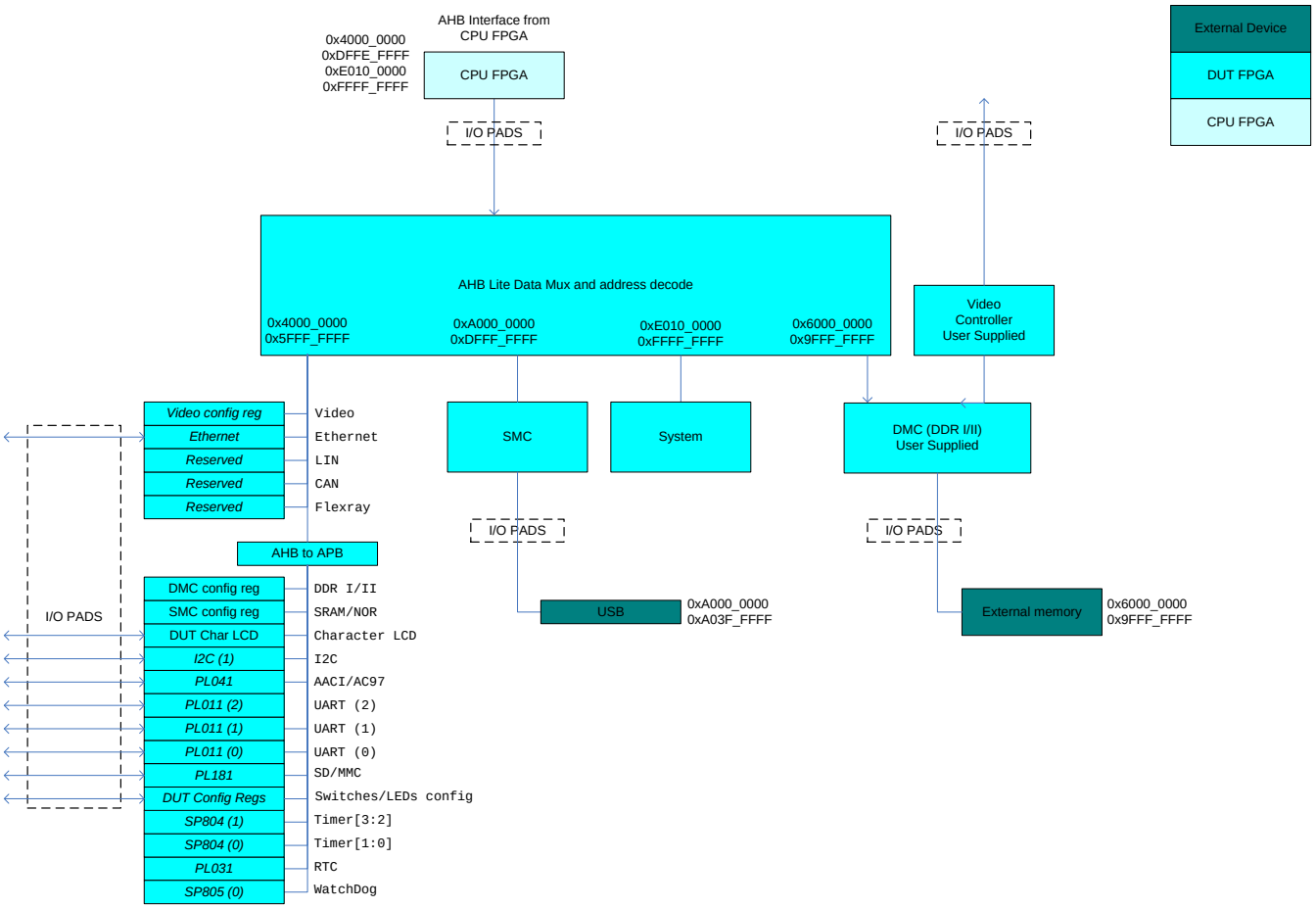


Figure 6: Bus Architecture of DUT FPGA

Copyright © 2009 ARM Limited. All rights reserved.

2.3 Clocks and Resets

Figure 7 below shows the Clock factory which is a simple switch matrix to select any of the input clock sources and route them to the output clocks for distribution to the CPU and DUT FPGAs.

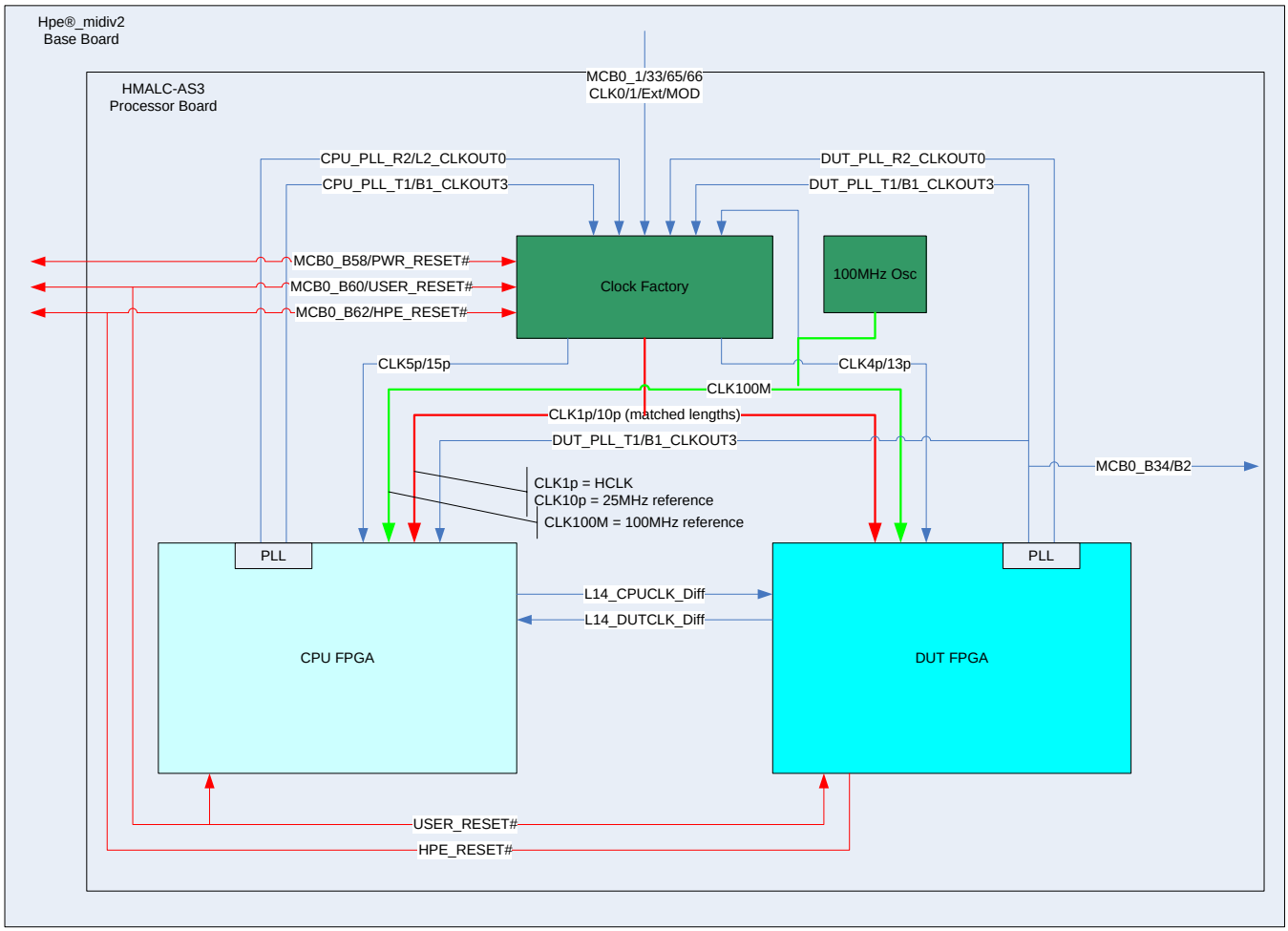


Figure 7: MPS Clock and Reset Architecture

The user has control of the input clocks from the DUT FPGA (e.g. DUT_PLL_R2_CLOCKOUT0) and can use PLLs within the FPGA to set operating frequencies and drive these signals to the clock factory. The clock factory can then route these signals back to either or both the CPU and DUT FPGA (e.g. CLK15p) to allow changes of clock frequency etc on FPGA clock input pins.

Copyright © 2009 ARM Limited. All rights reserved.

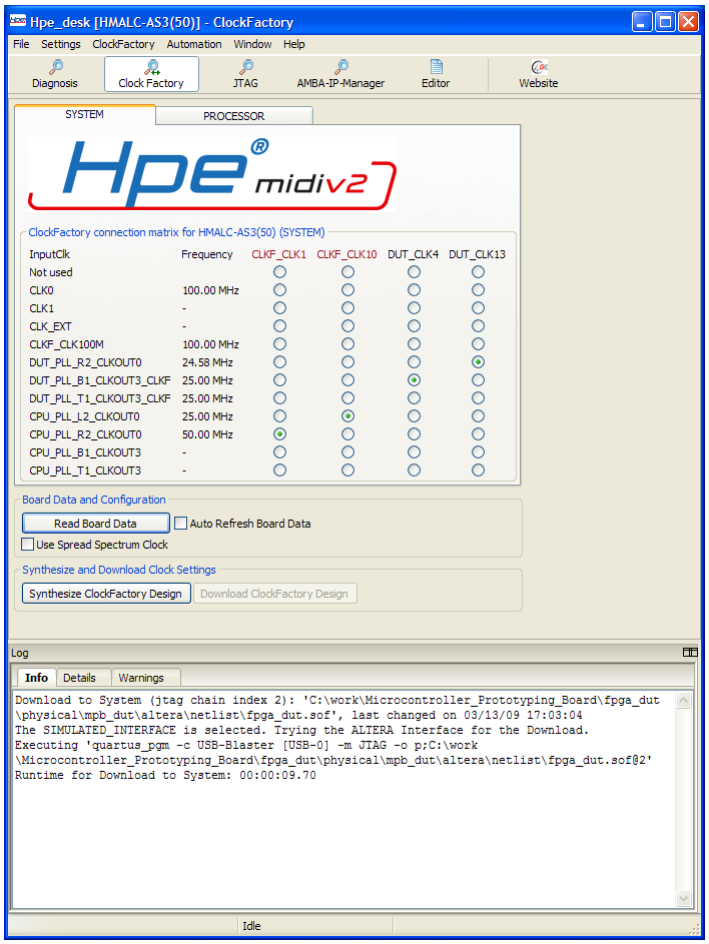


Figure 8: Hpe_desk clock factory configuration

The configuration of the clock factory is performed using Hpe_desk and is a simple button selection matrix to define the routing of clocks. Figure 8 shows the selection of source clock CPU_PLL_R2_CLKOUT0 (from CPU FPGA) driving the destination clock CLKF_CLK1 back into both CPU and DUT FPGA.

CLKF_CLK100M is the default 100MHz clock source and should be used as the reference for all PLL generated clocks.

2.3.1 Clock Routing

Name	Source	Destination	Note
CLK100M	Osc	CF,DUT,CPU	Buffered 100MHz clock output to DUT and CPU
CLK0	BB	CF	Oscillator module on Baseboard
CLK1	BB	CF	Oscillator module on Baseboard
EXT	BB	CF	External SMB clock input on Baseboard
CLK5p	CF	CPU	Direct connection to CPU only
CLK15p	CF	CPU	Direct connection to CPU only
CLK1p	CF	CPU,DUT	Buffered match lengths to DUT & CPU (HCLK)
CLK10p	CF	CPU,DUT	Buffered match lengths to DUT & CPU (CLK25MHz)
CLK4p	CF	DUT	Direct connection to DUT only
CLK13p	CF	DUT	Direct connection to DUT only
DUT_PLL_T1_CLKOUT3	DUT	CF,CPU,MCB0	Buffered FPGA output from internal PLL
DUT_PLL_B1_CLKOUT3	DUT	CF,CPU,MCB0	Buffered FPGA output from internal PLL
DUT_PLL_R2_CLKOUT0	DUT	CF	Direct FPGA output from internal PLL
CPU_PLL_L2_CLKOUT0	CPU	CF	Direct FPGA output from internal PLL

Copyright © 2009 ARM Limited. All rights reserved.

Name	Source	Destination	Note
CPU_PLL_R2_CLKOUT0	CPU	CF	Direct FPGA output from internal PLL
CPU_PLL_B1_CLKOUT3	CPU	CF	Direct FPGA output from internal PLL
CPU_PLL_T1_CLKOUT3	CPU	CF	Direct FPGA output from internal PLL
L14_CPUCLK_Diff	CPU	DUT	Direct FPGA differential output for L14 interface
L14_DUTCLK_Diff	DUT	CPU	Direct FPGA differential output for L14 interface

Table 1: Clock Routing

2.3.2 Reset Routing

Name	Source	Destination	Note
PWR_RESET#	BB	CF	O/D output from supply monitor (internal use)
USER_RESET#	All	CF,CPU,DUT	Push button on Processor board O/D
HPE_RESET#	CF	BB, DUT	Driven by USER_RESET# and PWR_RESET#

Table 2: Reset Routing

Notes:

BB: BaseBoard

CF: Clock Factory

CPU: CPU FPGA

DUT: DUT FPGA

OSC: Crystal Oscillator module.

The System only uses the USER_RESET# signal and this drives all internal resets (nPOR, nHRESET etc). The design ignores PWR_RESET# and HPE_RESET#.

The CPU FPGA drives the nHRESET signal between the CPU and DUT FPGA to create a synchronous reset (with respect to HCLK) in the DUT FPGA. The DUT FPGA uses this to resynchronise resets to all other clock domains within the FPGA.

Copyright © 2009 ARM Limited. All rights reserved.

3 Programmer's Model

3.1 Interrupts

The interrupt controller is integrated into the processor and resides in the CPU FPGA. Details of the architecture for this can be found in the relevant processor Application Note and documentation. The mapping of the example system peripherals to interrupts is irrespective of the controller implementation and detailed here. Figure 8 describes which interrupt is driven by each peripheral. An NMI input is available on some processors and is also shown below.

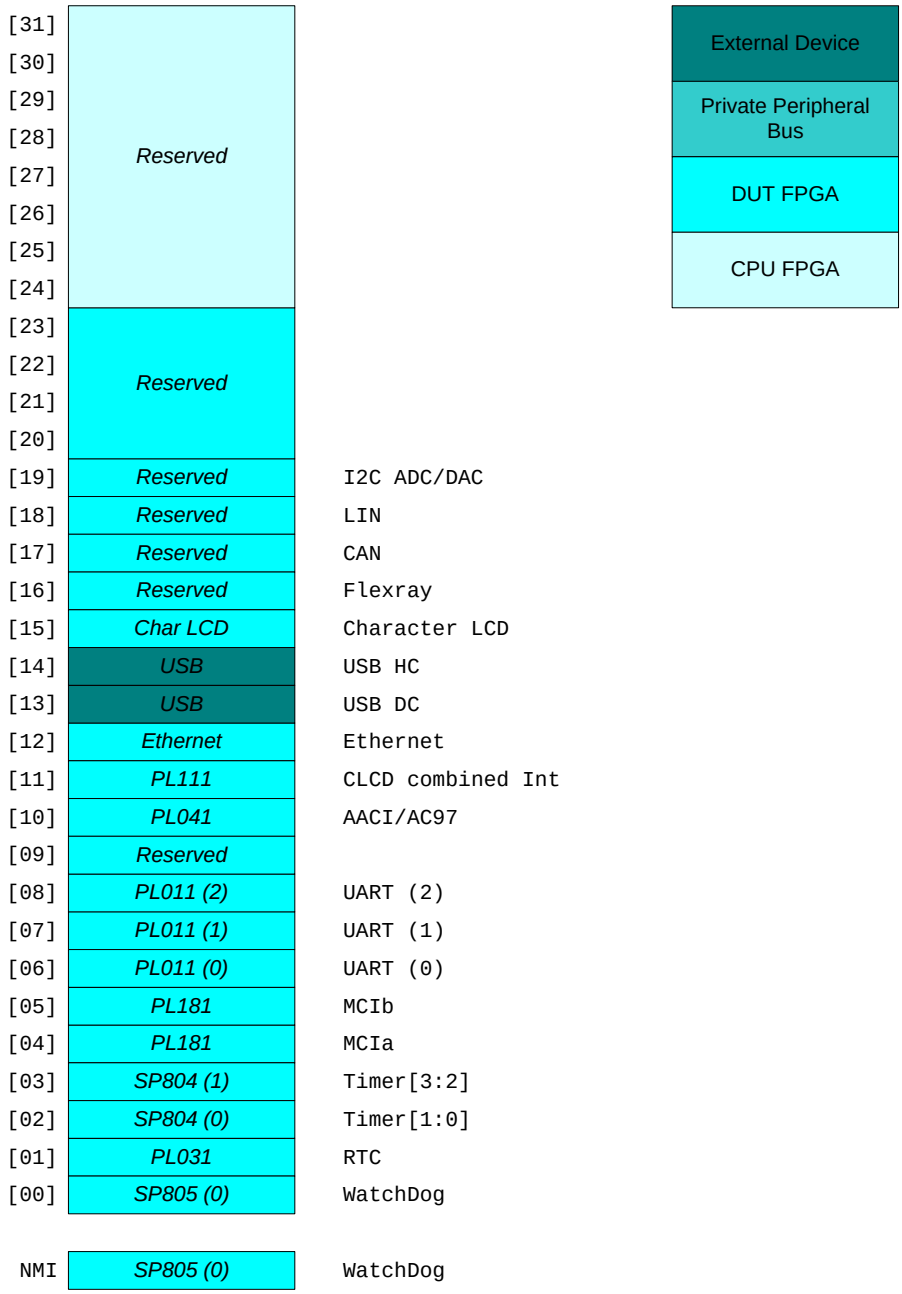


Figure 9: **Interrupt Allocation Table**

Copyright © 2009 ARM Limited. All rights reserved.

3.2 Memory map

Figure 10 details the memory ranges available to the DUT FPGA and the assignment of memory locations for the example system.

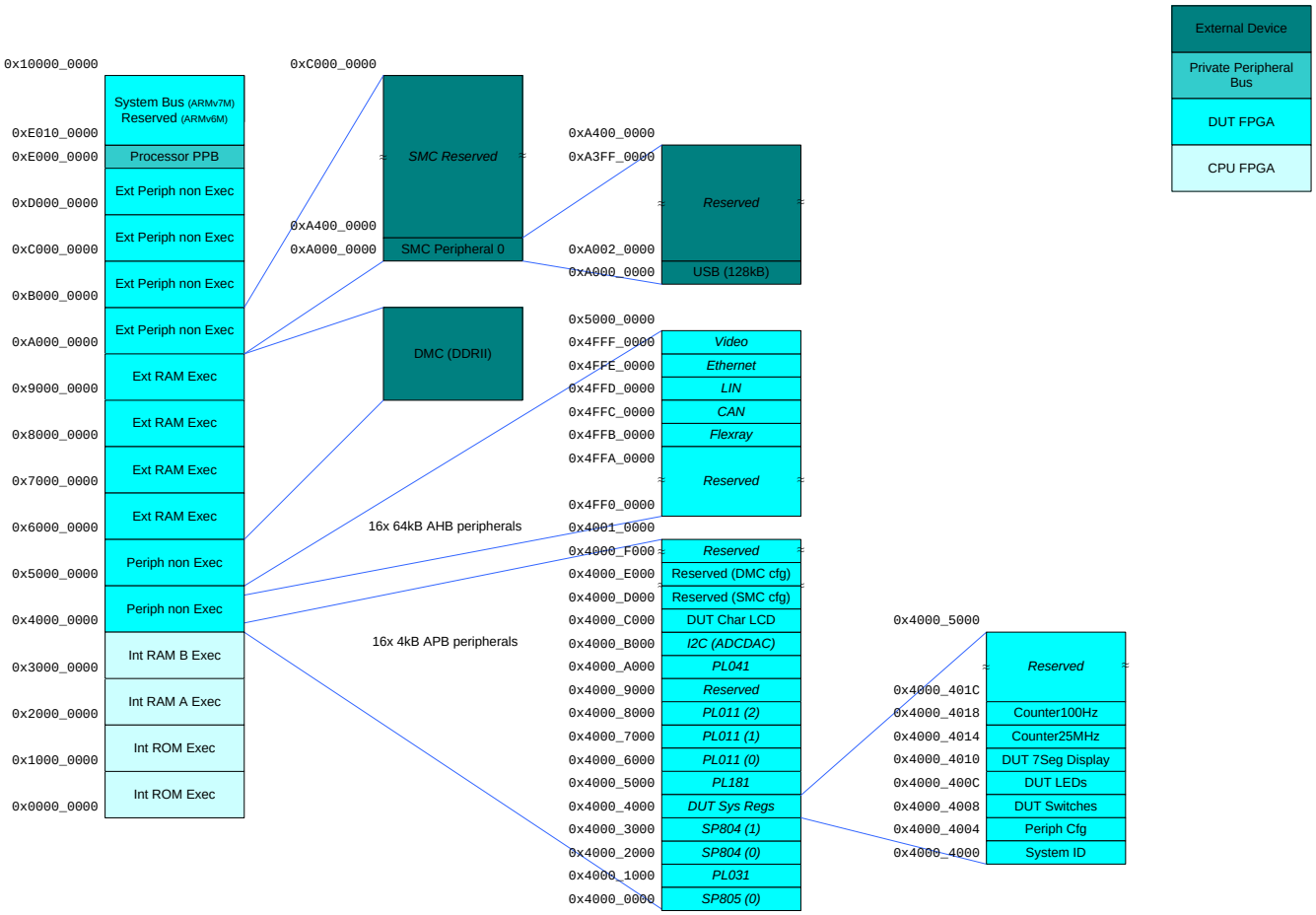


Figure 10: DUT FPGA memory map

3.3 DUT FPGA Registers

3.3.1 System Registers

The DUT specific registers are mapped to a 16KB area at 0x4000_4000. The addresses in this section are all relative to this base address.

Register	Offset	Access	Reset	Note
SYS_ID	'h0000	RO	'h102304xx	Board and FPGA identifier
SYS_PERCFG	'h0004	RW	'h00000000	Peripheral control signals
SYS_SW	'h0008	RO	'h000000xx	Indicates user switch settings
SYS_LED	'h000C	RW	'h00000000	Sets LED outputs.
SYS_7SEG	'h0010	RW	'h00000000	Sets LED outputs.
SYS_CNT25MHz	'h0014	RO	'h00000000	Free running counter incrementing at 25MHz
SYS_CNT100Hz	'h0018	RO	'h00000000	Free running counter incrementing at 100Hz

3.3.1.1 ID register (SYS_ID)

Name	Bits	Access	Reset	Note
REV	31:28	RO	'h1	Board Revision B
BOARD	27:16	RO	'h023	HBI Board number

Copyright © 2009 ARM Limited. All rights reserved.

VARIANT	15:12	RO	'h0	Build Variant of board
ARCH	11:8	RO	'h4	Bus Architecture (4 AHB)
BUILD	7:0	RO	'hxx	FPGA build

3.3.1.2 Peripheral configuration (SYS_PERCFG)

Name	Bits	Access	Reset	Note
Reserved	31:12			
USB_FORCE_SLOW	11	RW	'b0	Forces the USB interface to operate slowly
HUMI_MODE	10:8	RW	'b000	Operation mode of HUMI multiplexer (see table)
USB_HC_WAKE	7	RO	'b-	Status of USB Host Controller Wake/Suspend signal
USB_DC_WAKE	6	RO	'b-	Status of USB Device Controller Wake/Suspend signal
Reserved	5	RO	'b0	
Reserved	4	RW	'b0	
Reserved	3	RW	'b0	
Reserved	2	RO	'b0	
WPROT	1	RO	'b0	Status of MCI WPROT bit, 1 write protected
CARDIN	0	RO	'b0	Status of MCI card Present, 1 card inserted

The Human Interface (HUMI) Mode bits define how the scheduler selects the different display components on the system. This can be used for system debug.

Mode	Bit value	Note
Scheduler	000	Round robin schedule to all HUMI devices
LEDs	001	HUMI LEDs only output
7Segment 0	010	HUMI 7Segment display 0 only output
7Segment 1	011	HUMI 7Segment display 1 only output
7Segment 2	100	HUMI 7Segment display 2 only output
7Segment 3	101	HUMI 7Segment display 3 only output
Character LCD	110	HUMI character LCD only output
Reserved	111	Reserved - Do Not Use

3.3.1.3 Switches (SYS_SW)

Name	Bits	Access	Reset	Note
Reserved	31:8			
USER_BUT[3:0]	7:4	RO	'h-	Always returns value of user buttons
USER_SW[3:0]	3:0	RO	'h-	Always returns value of user switches

3.3.1.4 LEDs (SYS_LED)

Name	Bits	Access	Reset	Note
Reserved	31:8			
LED	7:0	RW	'h00	Returns value in register. 1 is LED on 0 LED off

3.3.1.5 Display output (SYS_7SEG)

Name	Bits	Access	Reset	Note
DISP3	31:24	RW	'h00	Segments for display 3
DISP2	23:16	RW	'h00	Segments for display 2
DISP1	15:8	RW	'h00	Segments for display 1
DISP0	7:0	RW	'h00	Segments for display 0

Copyright © 2009 ARM Limited. All rights reserved.

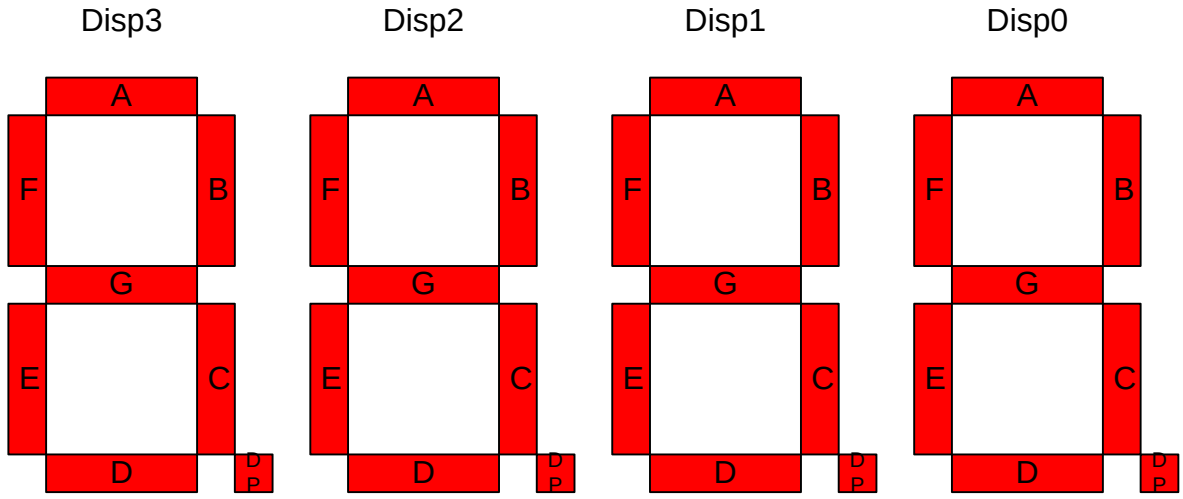


Figure 11: 7 Segment Display Segment Identification

Name	Bit	Note
DP	7	1 is Decimal Point on, 0 is Decimal Point off.
G	6	1 is segment on, 0 is segment off.
F	5	1 is segment on, 0 is segment off.
E	4	1 is segment on, 0 is segment off.
D	3	1 is segment on, 0 is segment off.
C	2	1 is segment on, 0 is segment off.
B	1	1 is segment on, 0 is segment off.
A	0	1 is segment on, 0 is segment off.

3.3.1.6 25MHz Counter (SYS_CNT25MHz)

Name	Bits	Access	Reset	Note
Count25MHz	31:0	RO	'h00000000	Free running counter from 25MHz clock

3.3.1.7 100Hz Counter (SYS_CNT100Hz)

Name	Bits	Access	Reset	Note
Count100Hz	31:0	RO	'h00000000	Free running counter from 100Hz clock

3.3.2 ADC/DAC (I^2C)

The DS702 peripheral is used for the interface and implements a bit banging method for the I^2C interface. The base address for this peripheral is 0x4000_B000. Refer to [2] for further details.

Register	Offset	Access	Reset	Note
SB_CONTROL	'h0000	R	'b0-	Status Register of I/O signals
SB_CONTROLS	'h0000	W	'b00	Set Output bits
SB_CONTROLC	'h0004	WO	'b00	Clear Output bits

3.3.2.1 SB Status register (SB_CONTROL)

Name	Bits	Access	Reset	Note
Reserved	31:2			
SB_SDA (data wire)	1	RO	'b0	Level of SDA signal

Copyright © 2009 ARM Limited. All rights reserved.

SB_SCL (clk wire)	0	RO	'b0	Level of SCL signal

3.3.2.2 SB Set register (SB_CONTROLS)

Name	Bits	Access	Reset	Note
Reserved	31:2			
SB_nSDAOUTEN	1	W	'b0	Sets SDA line when 1
SB_SCLOUT	0	W	'b0	Sets SCL line when 1

3.3.2.3 SB Clear register (SB_CONTROLC)

Name	Bits	Access	Reset	Note
Reserved	31:2			
SB_nSDAOUTEN	1	W	'b0	Clears SDA line when 1
SB_SCLOUT	0	W	'b0	Clears SCL line when 1

3.3.2.4 Basic Timing

The basic I^2C (Inter-Integrated Circuit) timing diagram is shown in Figure 11.The ACK is a returned value from the target device (responding to a data burst being received).

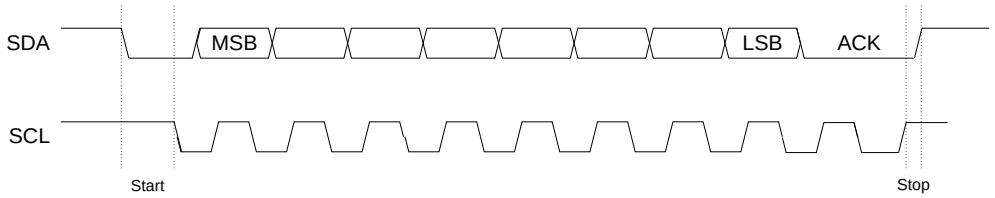


Figure 12: Timing Diagram of the I^2Cbus

3.3.3 Character LCD

The Character display component is the DS700 and interfaces to the industry standard Hitachi HD44780 controller. It uses 11 signals 8 data, 1 strobe (E), read/write (RnW) and Register/data select (RS).

Note: the interface can be a 4-bit or 8-bit interface. For this application it is in 8-bit mode.

Note: When the display is used with a 4-bit interface an 8-bit value has to be written/read as two consecutive nibbles, writing/reading bits [7:4] first into register bits [7:4], then writing/reading bits [3:0] into register bits [7:4].

Register	Offset	Access	Reset	Note
CHAR_COM	'h0000	RW	'h00000000	A write will write to the display controller command register. A read will initiate a status register access (returns value later in CHAR-RD).
CHAR_DAT	'h0004	RW	'h00000000	A write will write to the display controller data register. A read will initiate a data register access (returns value later in CHAR-RD).
CHAR_RD	'h0008	RO	'h00000000	Contains data from last CHAR_COM or CHAR_DAT read when CHAR_RAW[8] is set.
CHAR_RAW	'h000C	RW	'h00000000	Reading bit 8 indicates if access is complete. Writing 0 to bit 8 clears bit.
CHAR_MASK	'h0010	RW	'h00000000	Set bit 0 to 1 will generate interrupt when access completes.
CHAR_STAT	'h0014	RO	'h00000000	Returns status of Access Complete ANDed with CHAR MASK.

3.3.3.1 Character Command Register (CHAR_COM)

Copyright © 2009 ARM Limited. All rights reserved.

Name	Bits	Access	Reset	Note
Reserved	31:8	RW	'h00000000	
COMMAND	7:0	RW	'h00000000	A write will write to the display controller command register. A read will initiate a status register access (returns value later in CHAR_RAW and CHAR-RD).

3.3.3.2 Character Data Register (CHAR_DAT)

Name	Bits	Access	Reset	Note
Reserved	31:8	RW	'h00000000	
DATA	7:0	RW	'h00000000	A write will write to the display controller data register. A read will initiate a data register access (returns value later in CHAR_RAW and CHAR-RD

3.3.3.3 Character RD Register (CHAR_RD)

Name	Bits	Access	Reset	Note
Reserved	31:8	RO	'h00000000	
READ	7:0	RO	'h00000000	Contains data from last CHAR_COM or CHAR_DAT read when DONE is set.

3.3.3.4 Character Command Register (CHAR_RAW)

Name	Bits	Access	Reset	Note
Reserved	31:9	RW	'h0000000	
DONE	8	RW	'b0	Reading indicates if access is complete. Writing 0 clears bit.
Reserved	7:0	RW	'h00	

Note: If a transaction is attempted before DONE is asserted (CHAR_RAW register) by the controller then it may be ignored and the command/data transfer could be lost. Once DONE is asserted it can be cleared and a transaction started.

3.3.3.5 Character Interrupt Mask Register (CHAR_MASK)

Name	Bits	Access	Reset	Note
Reserved	31:1	RW	'h00000000	
MASKINT	0	RW	'b0	Set to 1 will generate interrupt when access completes (CHAR_DONE set)

3.3.3.6 Character Status Register (CHAR_STAT)

Name	Bits	Access	Reset	Note
Reserved	31:1	RW	'h00000000	
STATINT	0	RW	'b0	Returns status of CHAR_DONE ANDed with CHAR_MASKINT.

Copyright © 2009 ARM Limited. All rights reserved.

3.3.4 Video

This is a user supplied component with basic interfaces brought into the DUT FPGA to enable implementation.

The DVI-I controller device (Chrontel CH7302) uses an I^2C serial bus for configuration this is implemented in the CPU FPGA using the DS702.

The example clock source for the video pixel clock is derived from the PLL in the DUT FPGA and drives a clock input of the DUT FPGA (CLK4p) via the clock factory. The clock source is set by the Hpe_Desk application on the PC and the frequency is determined by the PLL implemented in the DUT FPGA.

For further data on the video encoding see the Datasheet for Chrontel CH7302 [7].

3.3.5 Timer

The SP804 ADK component is used for the timers. See the TRM for details about its functionality [8]. The Timer clock is set at 1MHz and is derived from the 100MHz clock.

The combined interrupt is used so each SP804 implementation only has 1 interrupt output (see section 3.1).

3.3.6 RTC

The PL031 PrimeCell is used as the RTC. See the TRM for details about its functionality [9]. The RTC clock is feed from the RTCCLK signal and is 1Hz and is derived from the 100MHz clock.

3.3.7 WatchDog

The SP805 ADK component is used for the watchdog. See the TRM for details about its functionality [10]. The clock is set at 1Hz and is derived from the 100MHz clock.

3.3.8 Dynamic Memory Controller

This is a user supplied component with basic interfaces brought into the DUT FPGA to enable implementation.

3.3.9 Static Memory Controller

The Static memory interface implemented in the design is specifically to allow communication to the ISP1761 USB device [14]. It does not require any configuration and is optimized for operation with a 50MHz AHB interface.

3.3.10 UARTs

The PL011 PrimeCell is used as the UART. See the TRM for details about its functionality [11].

The clock is set at 25MHz and is derived from the 100MHz clock.

3.3.11 Audio (AACI/AC97)

The PL041 PrimeCell is used as the AACI. See the TRM for details about its functionality [12]. The AACI is a modification of the PrimeCell with increased FIFO depth to help improve transfer performance in FPGA.

The clock source is derived from the baseboard and drives a clock input of the DUT FPGA (CLK13p). The FPGA can also drive the AC_EXT_CLK to set the clock, but this option is not implemented in the example system.

Copyright © 2009 ARM Limited. All rights reserved.

3.3.12 MMC/SD (MMCI)

The PL181 PrimeCell is used as the MMC/SD card controller. See the TRM for details about its functionality [13]. The MMCI uses the bits in the system registers to identify the write protection and card inserted status (see section 3.3.1 for details).

Copyright © 2009 ARM Limited. All rights reserved.

4 FPGA Design

All of the RTL for this design is provided as Verilog or precompiled netlists. Example files are provided to allow the system to be rebuilt with the Altera Quartus II tools. The readme files provided with the application note show the version of the tools used to build the design.

4.1 Directory structure

Figure 13: Top level directory structure

The directory structure is separated as follows:

- docs: Contains related documents including this document
- fpga_cpu: Contains precompiled encrypted images for the CPU FPGA in the design which contains the processor.
- fpga_dut: Contains the verilog RTL files which describe the structure and design of the example system in the DUT FPGA.
- software: bootmonitor and selftest example software.
- peripherals : Contains the verilog RTL or precompiled images of the peripherals used by the example design design In the DUT FPGA.

4.1.1 Peripherals

The peripherals used in the example system are stored in the peripherals directory which contains either verilog source or netlists. These files are read by the synthesis tool at run time and used to create a netlist for place and route. Each peripheral has its own directory under either logical (for Verilog source) or physical (for netlists).

Copyright © 2009 ARM Limited. All rights reserved.

```
logical
        ds700_charlcd
                verilog         Verilog source files
        ds702_i2c
                verilog         Verilog source files
        ds704_videogen
                verilog         Verilog source files
        pl011_uart
                verilog         Verilog source files
        pl022_ssp
                verilog         Verilog source files
        pl031_rtc
                verilog         Verilog source files
        sp804_timer
                verilog         Verilog source files
        sp805_watchdog
                verilog         Verilog source files
physical
        pl041_aaci
                synplify
                        netlist         Netlist files
        pl181_mmci
                synplify
                        netlist         Netlist files
```

Figure 14: peripherals directory structure

The description of each peripheral is outside the scope of this application note. The TRM for each peripheral can be found within each peripheral's folder and should be referenced for details about the operation and programmers model.

These peripheral files are supplied as an example for use in the example system.

4.1.2 DUT FPGA

The following directory structure shows the position of the files required to re-build the DUT FPGA image for the MPS.

Copyright © 2009 ARM Limited. All rights reserved.

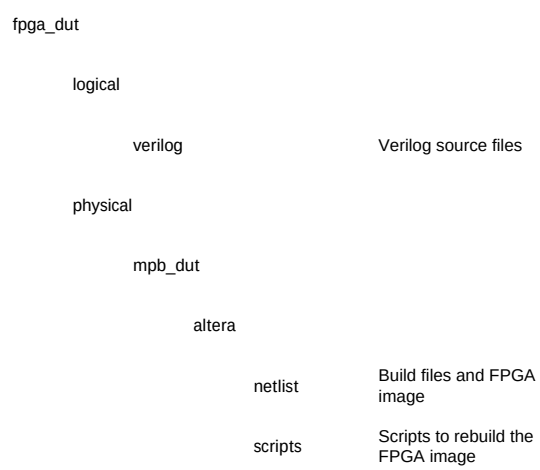


Figure 15: fpgu_dut directory structure

4.2 Re-building the DUT FPGA

4.2.1 Design file description

The fpga_dut directory contains the Verilog source required for the top-level of the example system. This directory together with the peripherals directory contains all the files required to build the example system. A description of each file is given in Table 3.

Filename	Note
AHB2APB.v	AHB to APB bridge
AHBAPBSys_dut.v	APB subsystem, instantiates the AHB to APB bridge, APB address decoder and APB peripherals. This requires changes to add or remove APB peripherals
AHBDecoder_dut.v	AHB address decoder. This requires changes to add or remove AHB peripherals
AHBDefaultSlave.v	Creates the error response if a unmapped AHB address is accessed
AHBMuxS2M.v	AHB multiplexer to direct the correct AHB peripheral access to the processor. This requires changes to add or remove AHB peripherals
AHBUSBController.v	AHB to 16bit static memory interface to access the USB controller
APBDecoder.v	APB address decoder. This requires changes to add or remove APB peripherals
APBRegs_dut.v	APB system registers used for system control. See section 3.3.1 for details
clock_divider.v	Generates peripheral clocks derived from 25MHz reference clock
dut_logic.v	AHB subsystem, instantiating the AHB peripherals and APB subsystem
fpga_dut.v	Top level, instantiating the AHB subsystem, Clock and reset logic, I/O functions (tri-state, I/O registers, PLL's etc) and Human Interface multiplexer
fpga_dut_defs.v	Defines for use by the design as compilation. See section 4.2.2 for details
humi_mux.v	Human Interface multiplexer to drive 7 segment displays, LEDS and Character LCD
pulse_gen.v	Pulse generator to create enable pulses for counters, deriving the pulse from the reference clock

Table 3: dut fpga verilog files

Also within this directory is the script for building the images to download to the FPGA. The synthesis script is in physical/MPS_dut/altera/scripts and produces a routed, placed design in the physical/MPS_dut/altera/netlist directory. See the Hpe_desk manuals for downloading this image to the FPGA [2].

4.2.2 Configuring the Example System

The example system has a definitions file (fpga_dut_defs.v) in the fpga_dut/logical/verilog directory to allow the selection of section of the design to be implemented or removed. It

Copyright © 2009 ARM Limited. All rights reserved.

also contains defines to determine the functionality of different sections of the design. The description of the `defines is given in Table 4: System Configuration.

Define	Note
`define ARM_MPS_INCLUDE_CLCD	This includes the Colour LCD and Video controller block (user supplied). Without this define, the pixel clock is not driven and the data and control lines are tied to '0', which means that the video bus will not drive any data or clock
`define ARM_MPS_VGA	Implement the VGA Pattern Generator 640x480 if no CLCD present
`define ARM_MPS_SVGA	Implement the VGA Pattern Generator 800x600 if no CLCD present
`define ARM_MPS_XVGA	Implement the VGA Pattern Generator 1024x768 if no CLCD present
`define ARM_MPS_INCLUDE_AACI	This includes the Primecell PL041 (Audio Codec interface).
`define ARM_MPS_INCLUDE_MCI	This includes the Primecell PL181 (MMC/SD interface).
`define ARM_MPS_INCLUDE_DMC	This includes the DDR memory interface (user supplied). If these components are not included then the video and LCD controller does not have any route to memory so the address out of the video and LCD controller block is connected to the read data bus giving a static image on any screen attached to that interface
`define ARM_MPS_DUT_SYS_ID_REG	This is the value shown in the SYS_ID register (see section 3.3.1.1for details)

Table 4: System Configuration

By default, the audio codec and MMC/SD card interfaces are implemented and the DMC and CLCD are not implemented. By un-commenting or commenting the `defines, the system configuration can be altered.

4.2.3 Building the FPGA image

To rebuild the image and download to the MPS the Altera Quartus II tools (version 8.1 or later) and the Hpe®_desk application need to be installed on the PC being used. Please refer to the MPS QuickStart guide for details on installing these.

Once the software is installed the example system FPGA image can be rebuilt by running a single batch script. This script invokes the Altera Quartus II tools to perform both synthesis and place and route functions. Once this script has completed, the FPGA image file (fpga_dut.sof) can be download to the DUT FPGA using the Hpe®_desk application.

The files in the scripts directory (fpga_dut\physical\MPS_dut\altera\scripts) perform the functions described in Table 5: FPGA scripts;

Filename	Note
Build.bat	Windows batch file to rebuild the FPGA image in the netlist directory
fpga_dut.qpf	Quartus II project file. This should not be altered
fpga_dut.qsf	Quartus II TCL script. It defines the FPGA pin assignment, instantiation of pullup or down resistors, files to be synthesised. This will require changes when modifying the DUT FPGA design and adding or removing peripherals
fpga_dut.sdc	Defines the timing constraints for the design and is configured to ensure operation with the example peripherals and processor (CPU FPGA). This will require changes when modifying the DUT FPGA design and adding or removing peripherals

Table 5: FPGA scripts

An explanation of the files contents (in Table 5: FPGA scripts) is beyond the scope of this application note and reference to the Altera Quartus II documentation is advised for details on the commands and structure of the files.

To build the FPGA image perform the following tasks;

1. Run the 'build.bat' file from the 'fpga_dut\physical\MPS_dut\altera\scripts' directory.
2. Check the FPGA image has been created by looking for the file 'fpga_dut.sof' in the 'fpga_dut\physical\MPS_dut\altera\netlist' directory.

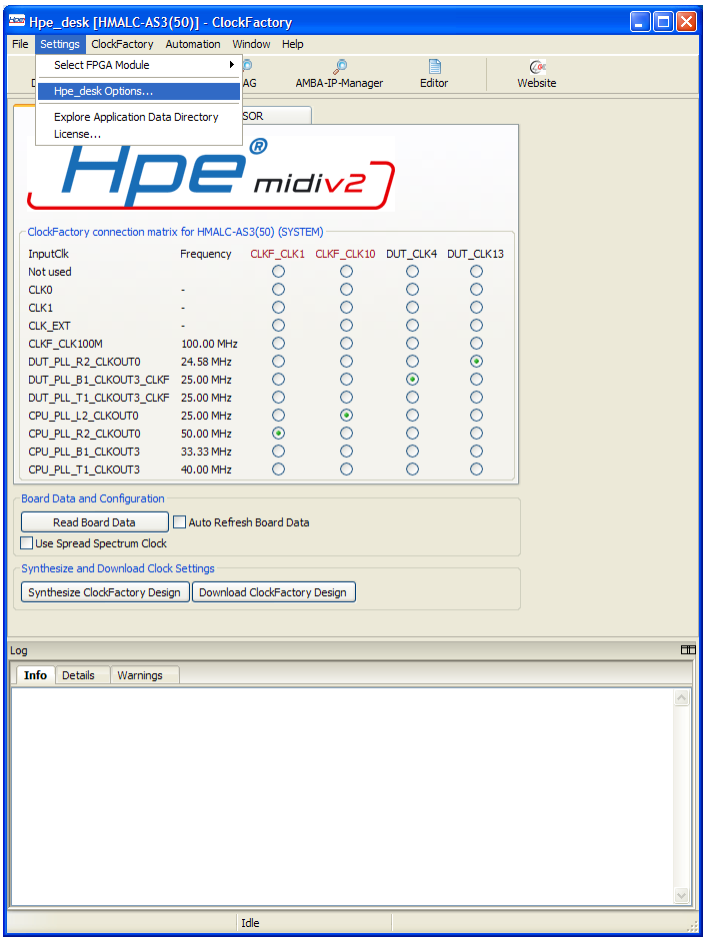
Copyright © 2009 ARM Limited. All rights reserved.

4.2.4 Configuring the MPS and Hpe®_desk

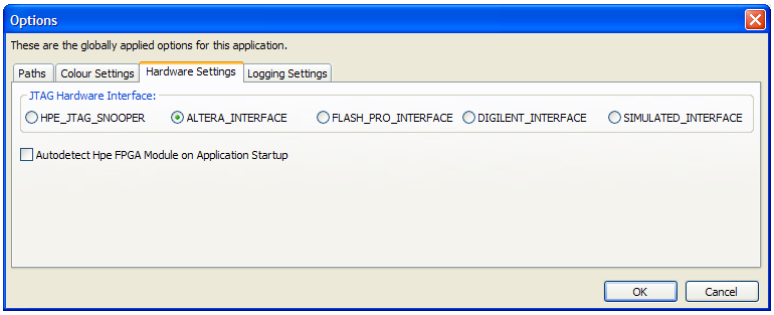
Before you can download the FPGA image to the MPS, the Hpe®_desk application needs to be installed and configured. Follow the following steps to configure;

1. Ensure the Hpe®_desk application has been installed.
2. Connect the PC via a USB cable to the USB OTG connector on the front of the MPS.
3. Power on the MPS.
4. Start the Hpe®_desk application.
5. Configure the connection method to the MPS

Select from the drop down menus, ‘Settings -> Hpe_desk Options…’



Select the ‘Hardware Settings’ tab from the ‘options’ box and then select the ‘ALTERA_INTERFACE’ radio button. Select the ‘OK’ button to complete configuration.

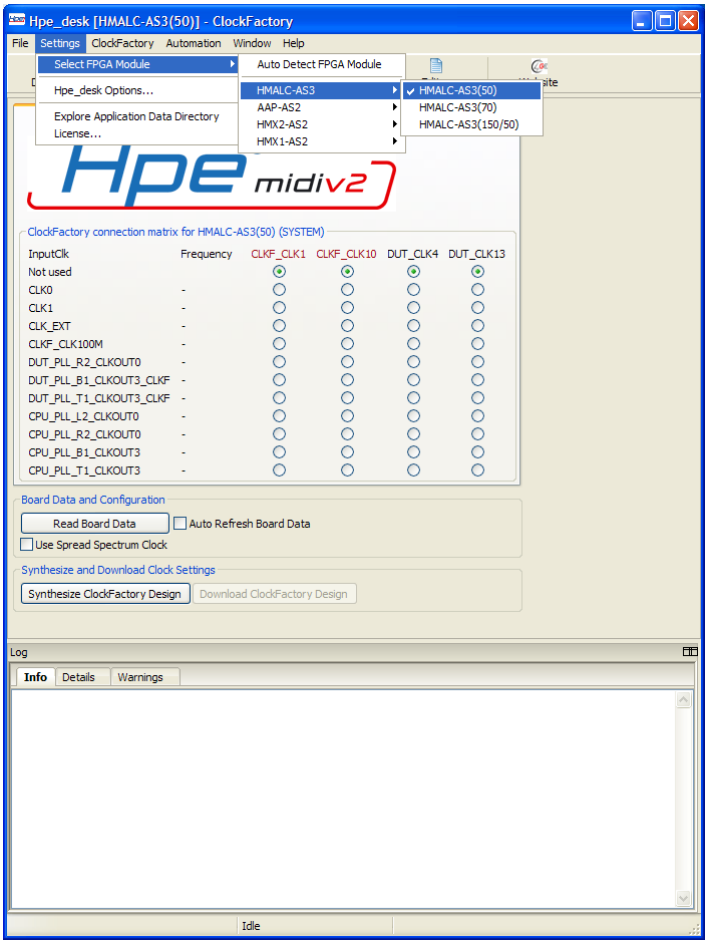


6. To configure the correct FPGA module for MPS.

Copyright © 2009 ARM Limited. All rights reserved.

Select from the drop down menu,

'Settings -> select FPGA module -> HMALC-AS3 -> HMALC-AS3(50)'



The Hpe®_desk application has now been configured to connect to the MPS and downloading of FPGA images can now be done.

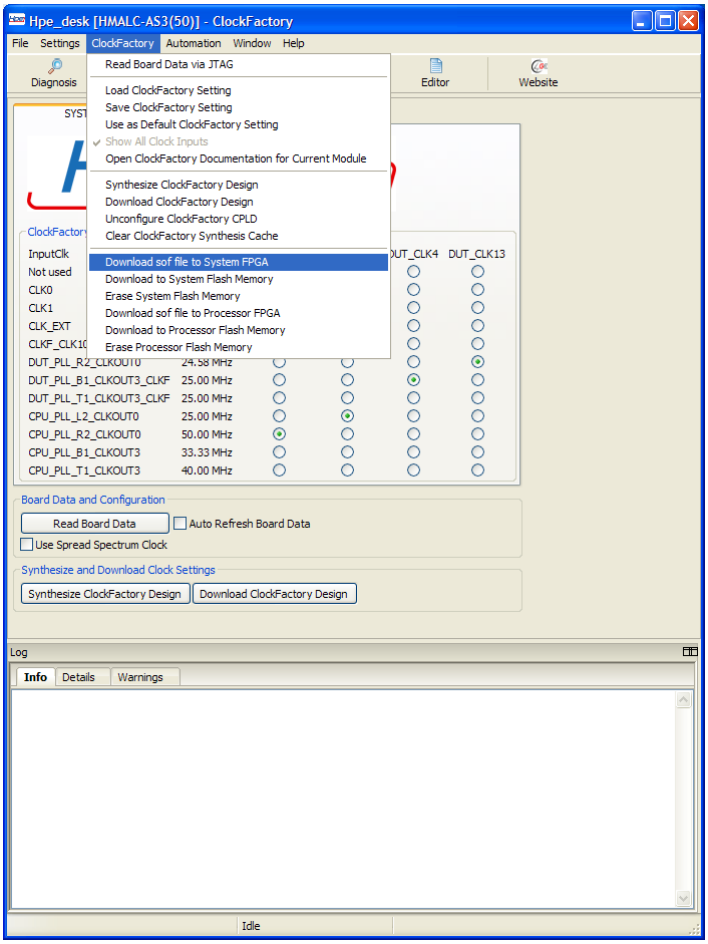
4.2.5 MPS FPGA image download (volatile)

Downloading directly to the FPGA is the fastest method to configure, but is a volatile download and once the power to the MPS is removed the downloaded image is lost and the non-volatile image is used on power up. It is recommended to try new designs using this method before committing to a non-volatile download.

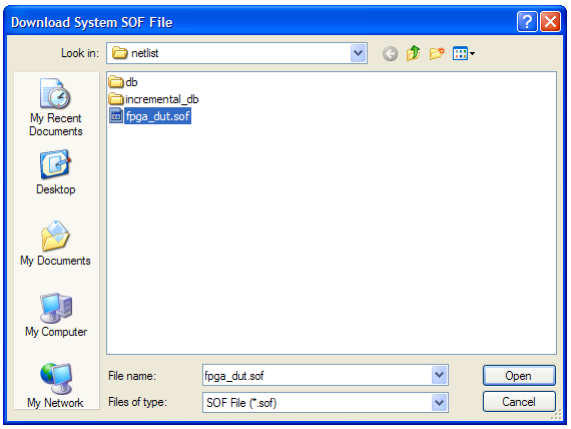
To download a volatile image to the MPS perform the following from the Hpe®_desk application;

Select from the drop down menus, 'ClockFactory -> Download sof file to System FPGA'

Copyright © 2009 ARM Limited. All rights reserved.



Navigate to and select the ‘fpga_dut.sof’ image from the netlist directory to download to the MPS.



The Hpe®_desk application will now directly program the FPGA with the image.

The Blue LED (FPGA CONFIG D) will go out while it is being configured and turn on once it is done and the text at the bottom of Hpe®_desk application (Log window) will inform you that it has configured. The DUT FPGA has now been updated with the new image.

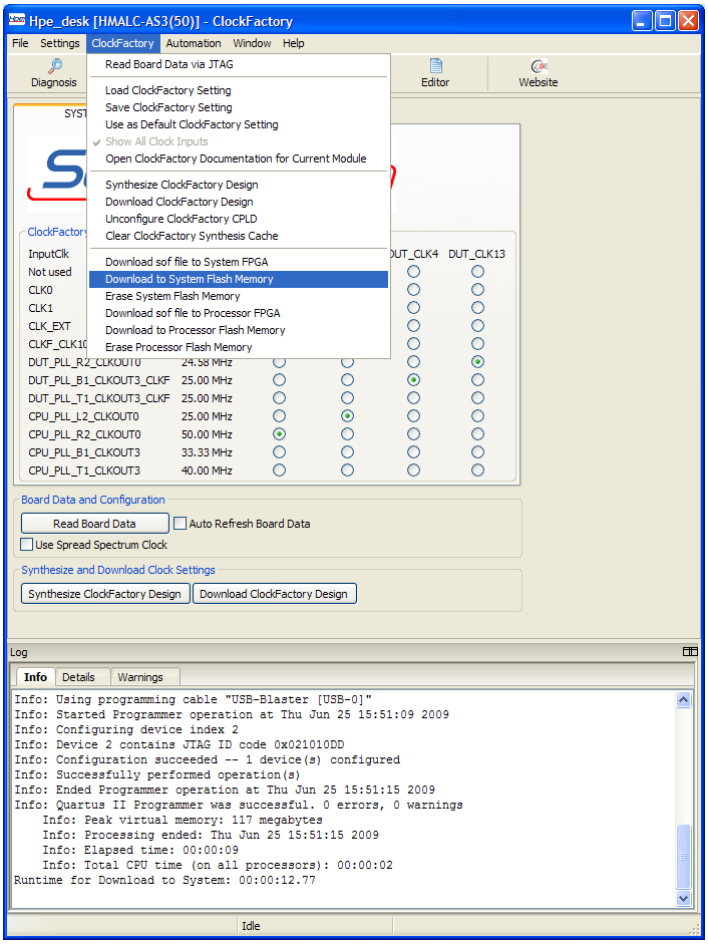
4.2.6 MPS FPGA image download (non-volatile)

Downloading to the Flash memory is a slow method, but is non-volatile and the image is reloaded on power up. It is recommended to use this when the design is stable.

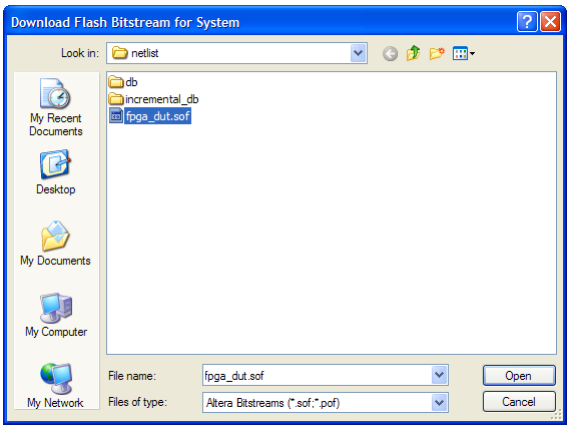
To download a non-volatile image to the MPS perform the following from the Hpe®_desk application;

Copyright © 2009 ARM Limited. All rights reserved.

Select from the drop down menu ‘ClockFactory -> Download to System Flash Memory’


Hpe_desk [HMALC-AS3(50)] - ClockFactory
File Settings ClockFactory Automation Window Help
Diagnosis
Editor
Website
Read Board Data via JTAG
Load ClockFactory Setting
Save ClockFactory Setting
Use as Default ClockFactory Setting
Show All Clock Inputs
Open ClockFactory Documentation for Current Module
Synthesize ClockFactory Design
Download ClockFactory Design
Unconfigure ClockFactory CPLD
Clear ClockFactory Synthesis Cache
Download sof file to System FPGA
Download to System Flash Memory
Erase System Flash Memory
Download sof file to Processor FPGA
Download to Processor Flash Memory
Erase Processor Flash Memory
InputClk
Not used
CLK0
CLK1
CLK_EXT
DUT_CLK4
DUT_CLK13
DUT_PLL_B1_CLKOUT3_CLKF 25.00 MHz
DUT_PLL_T1_CLKOUT3_CLKF 25.00 MHz
CPU_PLL_L2_CLKOUT0 25.00 MHz
CPU_PLL_R2_CLKOUT0 50.00 MHz
CPU_PLL_B1_CLKOUT3 33.33 MHz
CPU_PLL_T1_CLKOUT3 40.00 MHz
Board Data and Configuration
Read Board Data
Auto Refresh Board Data
Use Spread Spectrum Clock
Synthesize and Download Clock Settings
Synthesize ClockFactory Design
Download ClockFactory Design
Log
Info Details Warnings
Info: Using programming cable "USB-Blaster [USB-0]"
Info: Started Programmer operation at Thu Jun 25 15:51:09 2009
Info: Configuring device index 2
Info: Device 2 contains JTAG ID code 0x021010DD
Info: Configuration succeeded -- 1 device(s) configured
Info: Successfully performed operation(s)
Info: Ended Programmer operation at Thu Jun 25 15:51:15 2009
Info: Quartus II Programmer was successful. 0 errors, 0 warnings
Info: Peak virtual memory: 117 megabytes
Info: Processing ended: Thu Jun 25 15:51:15 2009
Info: Elapsed time: 00:00:09
Info: Total CPU time (on all processors): 00:00:02
Runtime for Download to System: 00:00:12.77
Idle

Navigate to and select the ‘fpga_dut.sof’ image from the netlist directory to download to the MPS.



The Hpe®_desk application will now directly program the FPGA with the image.

The Blue LED (FPGA CONFIG D) will go out while it is being configured and turn on once it is done and the text at the bottom of Hpe®_desk application (Log window) will inform you that it has configured. The DUT FPGA has now been updated with the new image.

4.3 Clock and reset settings

Please use the online help of the Hpe®_desk software for details on programming the clocks [2]. Section 2.3.1 gives a basic overview of the clock factory settings.

Copyright © 2009 ARM Limited. All rights reserved.

4.3.1 Clock and Reset Destinations

The following table gives the default clock frequency settings used by the example system and which clock inputs of the DUT FPGA are used.

Device	Clock Freq	Clock ref	Reset ref	Note
AHB/APB infrastructure	HCLK	CLK1p	HRESETn	Runs at CPU frequency
UART	25MHz	CLK10p	HRESETn	PL011
SPI	25MHz	CLK10p	HRESETn	PL022
I^2C	HCLK	CLK1p	HRESETn	DS702
VIDEO	25MHz	CLK4p	HRESETn	Video/LCD controller pixel clock
AC97	24.576MHz	CLK13p	HRESETn	PL041
SD/MMC	25MHz	CLK10p	HRESETn	PL181
Character LCD	HCLK	CLK1p	HRESETn	DS700
HUMI	500Hz	CLK1p	HRESETn	Multiplexer frequency
Timer	1MHz	CLK10p	HRESETn	SP804
Real Time Clock	1Hz	CLK10p	HRESETn	PL031
Watch Dog	1Hz	CLK10p	HRESETn	SP805
USB	HCLK	CLK1p	HRESETn	External IC ISP1761
Static Memory	HCLK	CLK1p	HRESETn	Memory controller on DUT FPGA

Table 6: Clock and Reset Destinations

All the peripherals that have an AHB or APB interface have that interface running at CLK1p.

CLK100M is used to derive all peripheral clocks where appropriate since this is a non variable clock and ideal for timers, watchdogs etc.

The Reset column refers to the reset signal that is re-synchronised to the respective clock domain.

Copyright © 2009 ARM Limited. All rights reserved.

5 Example Software

Under the MPS root directory you can find the software directory which contains the files required to rebuild the BootMonitor and Selftest software. It also contains the images for downloading to the MPS.

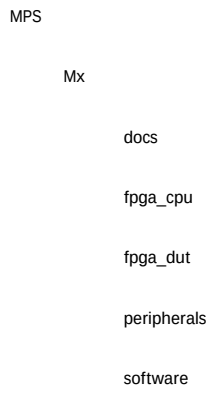


Figure 16 Top level directory structure

The directory structure is separated as follows:

- docs: Contains related documents including this document
- fpga_cpu: Contains precompiled encrypted images for the CPU FPGA in the design which contains the processor.
- fpga_dut: Contains the verilog RTL files which describe the structure and design of the example system in the DUT FPGA.
- software: BootMonitor and selftest example software.
- peripherals: Contains the verilog RTL or precompiled images of the peripherals used by the example design in the DUT FPGA.

5.1 BootMonitor

This is the application that runs when the system is booted, it handles the system initialization and has a command interpreter which accepts user input from the console to perform the following functions.

- Board configuration.
- General file operations using MM/SDCards configured as FAT16 8.3 filenames.
- Programming images into flash.
- Loading and running another application.
- Console supports semihosting via the debugger (not supported in MDK) or serial port.

It supports a single processor environment in little endian mode.

The BootMonitor reads the CPU User switches 0-2 (shown in 4) on power up and uses these to select the boot option. On delivery all the switches are set to ON and defaults to no boot script with auto detection of console interface, either semihosting (not supported on MDK) or UART port3 (shown in Figure 4).

SW1	SW2	SW3	Function	Note
ON	X	X	Normal boot	Use this as default

Copyright © 2009 ARM Limited. All rights reserved.

OFF	X	X	Run boot Script	This needs to be pre configured from the boot monitor command line
X	ON	ON	Auto Select between UART port3 and Semihosting for Console	Detects semihosting supported debugger
X	ON	OFF	Force UART port3 for Console	Always use UART port3 regardless of semihosting support
X	OFF	ON	Reserved	Do not use, undefined behaviour
X	OFF	OFF	Reserved	Do not use, undefined behaviour

Table 7: Bootmonitor switch utilisation

Note: no other switches are used by BootMonitor and are free for user use at power-up or reset.

When the MPS is turned on, Boot Monitor will start. The character display will show the Firmware (F/W) and Hardware (H/W) versions of the system. This is also output to the serial port for display on the terminal, if the switches are set correctly you will now enter the Boot Monitor command prompt on the terminal (SW[3:1] ON). The CPU LEDs (0 to 7) will cycle a lighted bit to show the Boot Monitor is running.

Pressing the reset button on the front panel will perform a hardware reset and the system will restart as if it had been power cycled.

5.1.1 Console via serial port

To use the serial port for the console interface to BootMonitor you will need to connect a serial null modem cable to RS232-4 (UART port3 above the power connector) and use a terminal emulation program (e.g. HyperTerminal) configured as 38,400 baud, 8bit data, no parity, 1 stop bit and no flow control to talk to the MPS.

5.1.2 Commands

5.1.2.1 Main Menu

Command Format	Note
ALIAS <alias> <command string>	Create an alias command <alias> for the string of commands in <command string>.
CD <directory path>	Change directory to the one specified in <directory path>.
CLEAR BOOTSCRIPT	Clear the current boot script. If no boot script is set then the boot monitor will always prompt for input no matter what the state of the 'run boot script' switch.
CONFIGURE	Enter Configure Submenu
CONVERT BINARY <binary-file> LOAD_ADDRESS <address> [ENTRY_POINT <address>]	Adds information required by the RUN command to execute a binary file. The command will produce a file with the same name as the specified binary file but with the '.exe' file extension.
COPY <file1> <file2>	Copies file <file1> to <file2>.
CREATE <file>	Create a file <file>.
DEBUG	Enter Debug Submenu
DELETE <file>	Delete a file <file>.

Copyright © 2009 ARM Limited. All rights reserved.

DIRECTORY [<directory>]	List files in <directory>.
DISPLAY BOOTSCRIPT	Display the current boot script
ECHO <text>	Prints string <text>.
EXIT	Exits the application or submenu.
FLASH	Enter Flash Submenu
HELP [<command>]	Provides help information on <command>. If <command> is not specified then all available commands are listed.
LOAD <image>	Loads image <image> into memory.
MKDIR <directory path>	Creates a new directory at the end of the given path
QUIT	Alias for 'EXIT'
RMDIR <directory path>	Removes a directory at the end of the given path
RENAME <file1> <file2>	Renames file <file1> to <file2>
RUN <image>	Load image <image> into memory and run it.
SDCARD	Enter SDCard Submenu
SET BOOTSCRIPT <script>	Set the current boot script. This script will be run at system reset if the run boot script switch is set.
TYPE <file>	Displays file <file>.

Table 8: Main Menu

Copyright © 2009 ARM Limited. All rights reserved.

5.1.2.2 Configure Submenu

Command Format	Note
DISPLAY DATE	Displays the current system date.
DISPLAY HARDWARE	Display hardware information
DISPLAY TIME	Displays the current system time.
EXIT	Exits the application or submenu.
HELP [<command>]	List commands
QUIT	Alias for 'EXIT'
RESET [IF_REQUIRED]	Resets this system. If the optional IF_REQUIRED qualifier is specified the system will only be reset if there has been a configuration change made that requires a reset.
SET BAUD <port #> <rate>	Sets UART <port #> to the specified <rate>. e.g. SET BAUD 0 9600 available ports 0-4
SET DATE <dd/mm/yy>	Sets system date in the form dd/mm/yy.
SET TIME <hh:mm:ss>	Sets system time in the form hh:mm:ss

Table 9: Configure Submenu

5.1.2.3 Debug Submenu

Command Format	Note
DEPOSIT <address> <value> [size]	Deposit value <value> to memory at <address>, optionally specifying the [size], it can be BYTE, HALFWORD or WORD (defaults to WORD).
DISABLE MESSAGES	Disables debug messages
ENABLE MESSAGES	Enables debug messages
EXAMINE <address> [<size>]	Examine memory at <address> for <size> number of bytes.
EXIT	Exit
GO <address>	Run code at <address>.
HELP [<command>]	List commands
MODIFY <address> <value> <mask> [size]	Performs a read/modify/write of memory at <address>, combining in with <value> which will be masked with <mask>. The size of the transfer can be optionally specified it can be BYTE, HALFWORD or WORD (defaults to WORD).
QUIT	Alias for 'EXIT'

Copyright © 2009 ARM Limited. All rights reserved.

START TIMER	Starts a timer which is stopped with the STOP TIMER command.
STOP TIMER	Stop a timer pervious started with the START TIMER command and displays elapsed time.

Table 10: Debug Submenu

5.1.2.4 Flash Submenu

Command Format	Note
DISPLAY IMAGE <name>	Display details of image <name>
ERASE IMAGE <name>	Erases image (or binary file) from flash.
ERASE RANGE <start_address> [<end_address>]	It is only possible to erase entire blocks of flash. Therefore the entire block of flash that contains <start_address>, the block that contains <end_address> and all intervening blocks will be erased. This may mean that data before <start_address> or after <end_address> will be erased if they are not on block boundaries. If the optional <end_address> parameter is not specified then only the single block of flash that contains <start_address> will be erased.
EXIT	Exit
HELP	List commands
LIST AREAS	List areas of flash, where an area is one or more contiguous blocks that are of the same size and use the same programming algorithms.
LIST IMAGES	List images in flash
LOAD <name>	Load image <name> from flash.
QUIT	Alias for 'EXIT'
RESERVE SPACE <address> <size>	Reserves space in flash for user applications that the boot monitor will not use.
RUN <name>	Load image <name> from flash and run it.
UNRESERVE SPACE <address>	Unreserves pervious reserved space in flash.

Copyright © 2009 ARM Limited. All rights reserved.

WRITE BINARY <file> [NAME <name>] [FLASH_ADDRESS <address>] [LOAD_ADDRESS <address>] [ENTRY_POINT <address>]	Writes a binary file to flash. The image will be identified in flash by a name derived from the filename, for example t:/images/boot_monitor.bin will be called boot_monitor, and this can be overridden by using the option NAME argument. You can specify where in flash the image is written by using the optional FLASH_ADDRESS argument (Note: if both FLASH_ADDRESS and LOAD_ADDRESS are specified and LOAD_ADDRESS is located in flash then LOAD_ADDRESS will be used and the FLASH_ADDRESS argument will be ignored). The optional LOAD_ADDRESS and ENTRY_POINT arguments allow you to specify these parameters, if ENTRY_POINT is not specified then to defaults to the load address.
WRITE IMAGE <file> [NAME <name>] [FLASH_ADDRESS <address>]	Writes an ELF image file to flash. The image will be identified in flash by a name derived from the file name, for example t:/images/boot_monitor.axf will be called boot_monitor, and this can be overridden by using the option NAME argument. You can specify where in flash the image is written by using the optional FLASH_ADDRESS argument (Note: if the image is linked to run from flash then this address will be used and the FLASH_ADDRESS argument will be ignored).

Table 11: Flash Submenu

5.1.2.5 SDCard Submenu

Command Format	Note
FORMAT [QUICK] [VOLUME <label>]	Formats the SDCard/MMC as FAT16 with 8.3 filenames QUICK: performs a quick format with only the FAT and bootsector updated. VOLUME <label> will add a Volume label to the disk as specified in the field <label>.
INFORM	Display details SD/MMC Card
INITIALISE	If the card has been changed use this command to re initialise it to determine it's features before using any other commands.
EXIT	Exit
HELP	List commands

Table 12: SDCard Submenu

5.1.3 Re-building the software

Currently, only a Windows development environment is supported. The firmware can be built using MDK and RVCT. Figure 16 **bootmonitor directory structure** shows the position of the executable files and scripts for rebuilding under both RVDS and MDK.

Copyright © 2009 ARM Limited. All rights reserved.

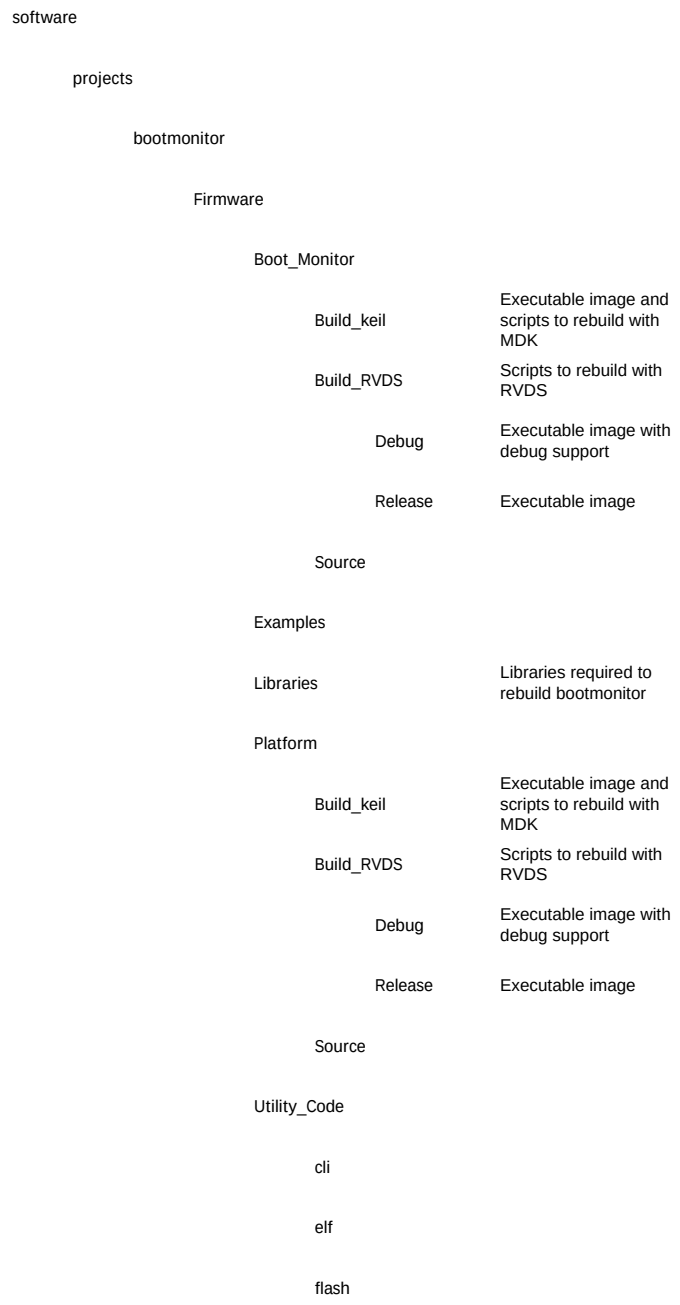


Figure 17: BootMonitor directory structure

5.2 Selftest

The selftest code allows the user to confirm the functionality of their MPS, and provides reference code for the example peripherals. The code tests the following peripherals: AACI, MMCI, USB, UARTs, character LCD, LEDs, switches, SRAM memory, RTC and system clocks/interrupts.

Selftest is designed to run on the MPS using MDK and RVDS. The user can interact with the software operation via the Serial Console for MDK or additionally the semihosting console window under RVDS.

The user interface displays a menu and prompts the user on how to operate each test. For more information on exactly how each test is working, refer to the provided source code, and readme files.

Copyright © 2009 ARM Limited. All rights reserved.

5.2.1 Re-building the software

Currently, only a Windows development environment is supported. Selftest can be built using MDK and RVCT. Figure 17 selftest directory structure shows the location of the executable files and scripts for rebuilding under both RVDS and MDK.

The project and executable files for selftest can be found under the \selftest\build\Build_Keil directory. To run all the tests it is necessary to connect a number of loopback cables to the board (for audio and UARTs).

After connecting the test harness and loading the program image into the debugger (selftest_MPS.axf) each of the MPS peripherals may be tested individually, or altogether using ‘Run all tests’. The tests perform register level and basic functional tests on the MPS hardware reporting any errors found.

The source code for the tests are brought together in a single project file \build\Build_Keil\selftest_MPS.Uv2.

The source code for each peripheral test is split into separate directories for example \apaaci\ contains apaaci.c and apaaci.h for testing the AACI peripheral. The \main\ folder contains main.c and common.c which provide the user menu and functions that are common to all peripheral tests.

Note: If the default install directory is not used, then the project will have to be rebuilt in order for the debugger to display the source code automatically.

Copyright © 2009 ARM Limited. All rights reserved.

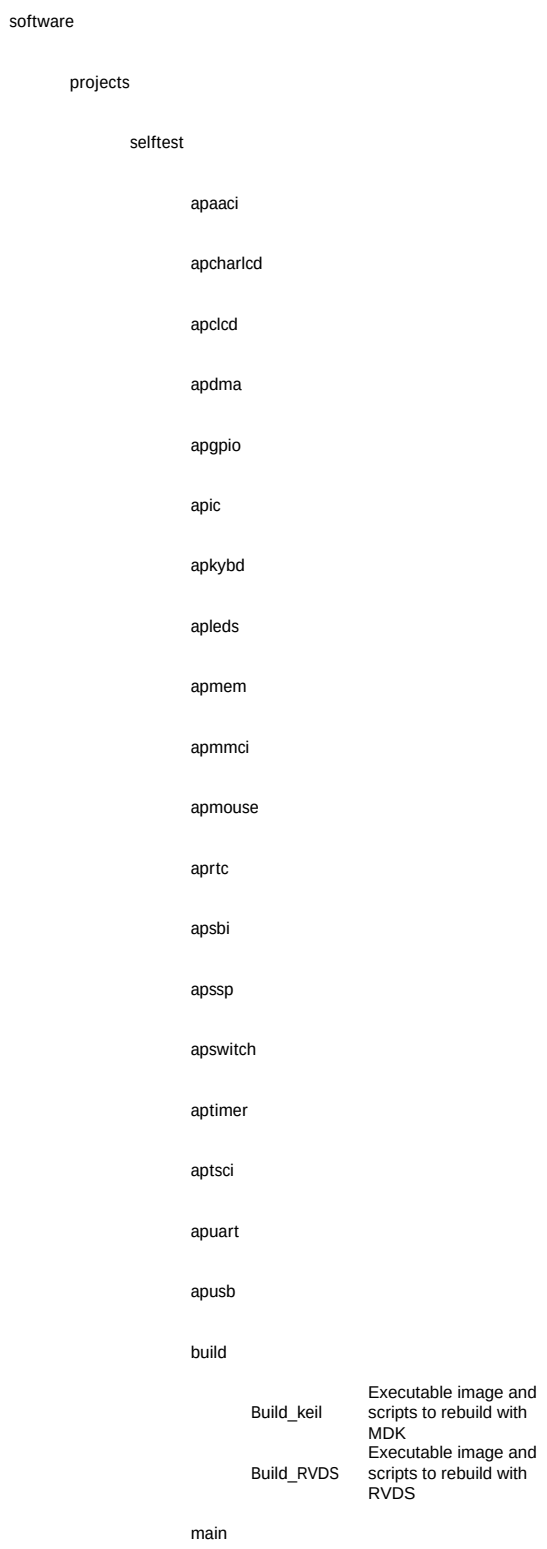


Figure 18: Selftest directory structure

5.2.2 Selftest test hardware

The MPS test code requires three separate cable assemblies to be connected to the board for complete testing. Note these cables are not supplied with the MPS but details of their connections are given here.

Copyright © 2009 ARM Limited. All rights reserved.

5.2.2.1 AACI loopback cable

The AACI test performs a loopback test from Line Level Out to Line Level In. This requires two 3.5mm stereo jack plugs which must all be wired as follows:

Connector A	Connector B
Tip	Tip
Ring	Ring
Screen	Screen

Table 13: AACI loopback cable

Connect the cable between the line in and line out sockets on the MPS (back panel).

5.2.2.2 UART loopback cable

The two UART cables have female 9-pin D-sub connectors on either end with connections as follows:

Connector A Pin	Connector A Name	Connector B Pin	Connector B name
1	N/C	1	N/C
2	RX	3	TX
3	TX	2	RX
4	DTR	6	DSR
5	GND	5	GND
6	DSR	4	DTR
7	RTS	8	CTS
8	CTS	7	RTS
9	N/C	9	N/C

Table 14: UART loopback cable

Connect one cable between the top two UART connectors and the other between the bottom two UART connectors one the MPS rear panel.

Copyright © 2009 ARM Limited. All rights reserved.

6 External Interfaces

This section shows the interfaces available on the Customer DUT FPGA.

The direction of the signals is shown from the point of view of the DUT FPGA, so an O signal goes from the DUT FPGA to the CPU FPGA (for example).

6.1 Clocks and Resets

Signal	Direction [Width]	Note
USER_RESETn	input	
HPE_RESETn	input	
DUT_CLK1	input	AHB system clock input
DUT_CLK4	input	Video Pixel Clock input
DUT_CLK10	input	Peripheral reference clock input
DUT_CLK13	input	AACI bit clock input (x2)
DUT_CLK100M	input	Reference 100MHz input
DUT_PLL_B1_CLKOUT3	output	25MHz reference clock from FPGA PLL
DUT_PLL_R2_CLKOUT0	output	AACI 24.576MHz bit clock x2 from FPGA PLL
DUT_PLL_T1_CLKOUT3	output	Video Pixel Clock from FPGA PLL

Table 15: Clocks and Resets

6.2 Processor Interface

This interface contains the 32bit AHB-Lite bus from the processor to the peripherals together with interrupts to the processor and sideband signals between the processor and peripherals.

FPGA Signal	Function	Note
FPGA_IC[31:0]	HWDATA[31:0]	
FPGA_IC[32]	HWRITE	
FPGA_IC[35:33]	HBURST[2:0]	
FPGA_IC[36]	HMASTLOCK	
FPGA_IC[40:37]	HPROT[3:0]	
FPGA_IC[43:41]	HSIZE[2:0]	
FPGA_IC[45:44]	HTRANS[1:0]	
FPGA_IC[46]	HSEL	Not used, driven high by the CPU FPGA
FPGA_IC[78:47]	HADDR[31:0]	
FPGA_IC[110:79]	HRDATA[31:0]	
FPGA_IC[111]	HRESP	
FPGA_IC[112]	HREADY	
FPGA_IC[113]	HRESETn	
FPGA_IC[125:114]	INT[11:0]	Interrupt signals to processor
L14_DUTOUT_DN[11:0]	INT[23:12]	Interrupt signals to processor
L14_DUTOUT_DN[12]	NMI	Non maskable interrupt to processor
L14_CPUOUT_DN[0]	SLEEPING	
L14_CPUOUT_DN[1]	SLEEPDEEP	
L14_CPUOUT_DN[2]	HALTED	
L14_CPUOUT_DN[3]	LOCKUP	
L14_CPUOUT_DN[4]	TXEV	
L14_CPUOUT_DN[5]	RXEV	
L14_CPUOUT_DN[6]	WDOGRES	
L14_CPUOUT_DN[12:7]	N/C	
L14_CPUOUT_DP[9:0]	N/C	
L14_CPUOUT_CLK	N/C	

Table 16: Processor interface

Copyright © 2009 ARM Limited. All rights reserved.

6.3 Video Interface

The interface to the Video and LCD displays is driven by the processor FPGA, although the peripheral is in the DUT FPGA. The CPU FPGA acts as a pass through for the video/LCD signals and will also de-multiplex the data for the LCD display.

FPGA Signal	Function	Note
L14_DUTOUT_DP[11:0]	Video RGB[11:0]	DDR 24bit RGB video signals clocked by VIDEOCLK
L14_DUTOUT_DP[12]	Video Resetn	Video Reset active low
L14_DUTOUT_CLK	PCLK	Video Pixel clock
L14_CPUOUT_DP[10]	VSync	Video vertical sync pulse
L14_CPUOUT_DP[11]	HSync	Video horizontal sync pulse
L14_CPUOUT_DP[12]	DE	Video data enable strobe

Table 17: DUT to CPU FPGA video signals

The multiplexed DDR interface is used to drive the video data (via the CPU FPGA) to the video transmitter device (Chrontel CH7302).

The signals from the FPGA video inputs are mapped to the video outputs as follows.

Function	LCD name	Video Name	Note
PCLK	LCD_R_SHFCLK	VIDEOCLK	Video Pixel clock
HSync	LCD_R_HSYNC	VIDEOHSYNC	Video vertical sync pulse
VSync	LCD_R_VSYNC	VIDEOVSYNC	Video horizontal sync pulse
DE	LCD_R_M_DE	VIDEODE	Video data enable strobe
Video RGB[7:0]	-	VIDEO[7:0]	Red Data DDR encoded falling edge
Video RGB[11:8]	-	VIDEO[11:8]	Green Data DDR encoded rising and falling edge
Video RGB[7:0]	-	VIDEO[7:0]	Blue Data DDR encoded rising edge
Video RGB[7:2]	LCD_TTL_R[5:0]	-	Red Data MSB's (demux'ed by CPU FPGA)
Video RGB[11:8]	LCD_TTL_G[5:0]	-	Green Data MSB's (demux'ed by CPU FPGA)
Video RGB[7:0]	LCD_TTL_B[5:0]	-	Blue Data MSB's (demux'ed by CPU FPGA)
Video Resetn	-	VIDEORESETn	Video Reset active low
-	-	VIDEOHPINT	Hot Plug interrupt (Not Used)
-	-	VIDEOMODE	Video Mode GPIO pin of video chip (Not Used)
-	-	VIDEO_I2C_SCL	Video Chip configuration bus (controlled by CPU FPGA)
-	-	VIDEO_I2C_SDA	Video Chip configuration bus (controlled by CPU FPGA)
-	LCD_BLON	-	Back Light On (output tied to 1 on CPU FPGA)
-	LCD_VDON	-	LCD Power On (output tied to 1 on CPU FPGA)

Table 18: Video and LCD Connections

Copyright © 2009 ARM Limited. All rights reserved.

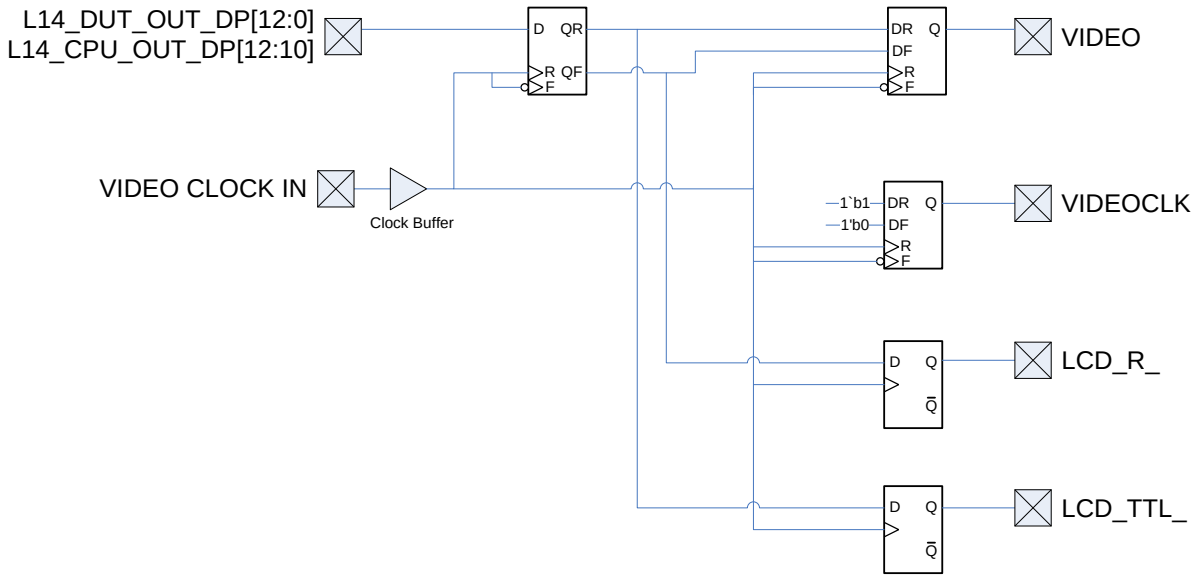


Figure 19: Video DDR interface

6.4 Human Interface

The Human Interface (HUMI) multiplexes the LED's, 7 Segment displays and character LCD together and drives them over the interface to the relevant devices.

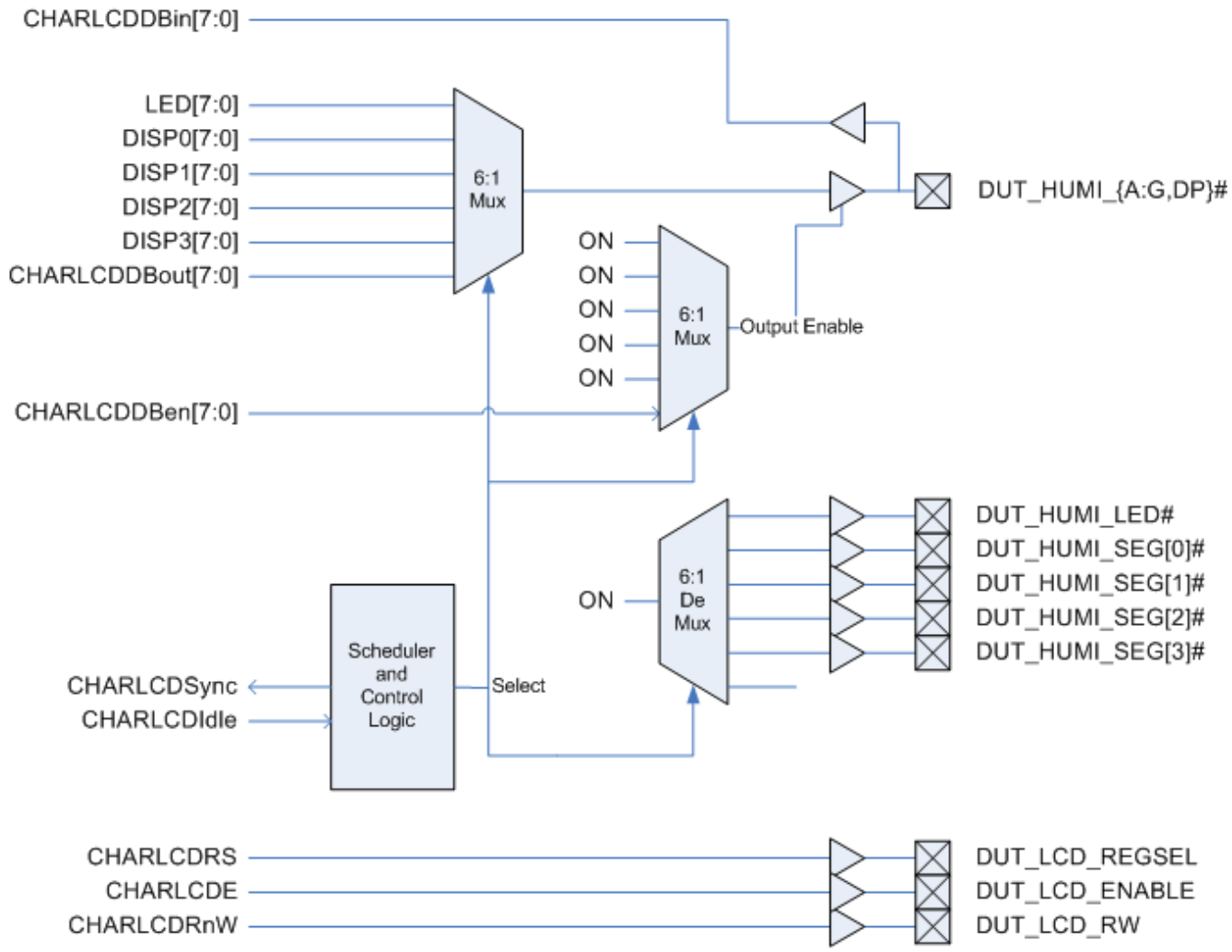
Signal	Direction [Width]	Note
BUTTON	input[4]	
DUT_DSW	input[4]	
DUT_HUMI_An	bi-dir	
DUT_HUMI_Bn	bi-dir	
DUT_HUMI_Cn	bi-dir	
DUT_HUMI_Dn	bi-dir	
DUT_HUMI_En	bi-dir	
DUT_HUMI_Fn	bi-dir	
DUT_HUMI_Gn	bi-dir	
DUT_HUMI_DPn	bi-dir	
DUT_HUMI_SEGn	output[4]	
DUT_HUMI_LEDn	output	
DUT_LCD_REGSEL	output	
DUT_LCD_RW	output	
DUT_LCD_ENABLE	output	

Table 19: Human Interface

Please refer to section 3.3 for details on the registers used for driving this interface. The buttons and switches can be read via the register SYS_SW (see section 3.3.1.3).

The values held in the SYS_LED and SYS_7SEG registers (see sections 3.3.1.4 and 3.3.1.5) are driven onto this interface based upon the value held in the HUMI_MODE field in the SYS_PERCFG register (see section 3.3.1.2).

Copyright © 2009 ARM Limited. All rights reserved.



Notes:
The Output Enable is driven at all times except when the Char LCD is selected, then it is controlled by the CHARLCDDBen signals.
The CHARLCDSync signal informs the CharLCD controller when it can perform the read and write transactions to the display, Enablde the DONE flag to be set.
The CHARLCDIdle signal informs the scheduler that a transaction is in progress and it must hold off switching until it completes
The scheduler must update each muxed output at minimum of 60Hz to ensure a flicker free image on 7 Segment displays and LEDs.

Figure 20: HUMI logic

The scheduler runs at a preset rate of 500Hz and selects a new driver for the data lines (DUT_HUMI_{A-G,DP}n) every 2 ms if the HUMI_MODE field is set to Scheduler (3'b000). The order of the cycle is:

- LEDs selected — the scheduler drives DUT_HUMI_LEDn low.
- Segment 0 — the scheduler drives DUT_HUMI_SEGn to 4'b1110.
- Segment 1 — the scheduler drives DUT_HUMI_SEGn to 4'b1101.
- Segment 2 — the scheduler drives DUT_HUMI_SEGn to 4'b1011.
- Segment 3 — the scheduler drives DUT_HUMI_SEGn to 4'b0111.

Copyright © 2009 ARM Limited. All rights reserved.

- Character LCD	the scheduler drives DUT_HUMI_SEGn to 4'b1110 and DUT_HUMI_LEDn high and enables the DUT_LCD_* interface control lines. Should any operation be in progress when the scheduler wishes to switch back to the LED, then change is halted whilst the operation completes and any new operation is prevented from starting (the CharLCD driver appears busy to the processor).

6.5 FPGA Configuration connections

These connections are not used by the design and are for FPGA configuration only.

Signal	Direction [Width]	Note
DUTFCP_CLK	input	
DUTFCP_DATA	input	
DUTFCP_PLTXT_RDY	input	
DUT_PORSEL	input	

Table 20: FPGA configuration Connections

6.6 SEMULATOR connections

The SEmulator connector is for use by the Gleichmann SEmulator product only. The signal assignments are shown for completeness only.

Signal	Direction [Width]	Note
SEDUT_L4_RXN	input[4]	
SEDUT_L4_RXP	input[4]	
SEDUT_L4_RXCLKN	input	
SEDUT_L4_RXCLKP	input	
SEDUT_L4_TXN	input[4]	
SEDUT_L4_TXP	input[4]	
SEDUT_L4_TXCLKN	input	
SEDUT_L4_TXCLKP	input	
SEDUT_RESETn	input	

Table 21: Semulator Connections

6.7 USB Interface

This interface is driven by a dedicated interface driver (16bit Static memory interface) which is designed to drive the NXP ISP1761 USB controller [14]. The interface driver is not configurable and is transparent to the rest of the system. All accesses to the USB device must be halfword accesses only.

Signal	Direction [Width]	Note
USB_A	output[17]	
USB_CSn	output	
USB_RDn	output	
USB_WRn	output	
USB_D	bi-dir[16]	
USB_DC_DACK	bi-dir	Not connected in this design.

Copyright © 2009 ARM Limited. All rights reserved.

USB_DC_DREQ	bi-dir	Not connected in this design.
USB_DC_IRQ	bi-dir	Used as input only. See interrupt table.
USB_DC_WAKEUPn	bi-dir	Not connected in this design.
USB_HC_DACK	bi-dir	Not connected in this design.
USB_HC_DREQ	bi-dir	Not connected in this design.
USB_HC_IRQ	bi-dir	Used as input only. See interrupt table.
USB_HC_WAKEUPn	bi-dir	Not connected in this design.

Table 22: USB Connections

6.8 Multi-Media/SD Card Interface

This interface is driven by the Primecell PL181 (see section 3.3.12) and is connected to the MM/SDCard connector on the rear panel.

Signal	Direction [Width]	Note
SD_SCLK	output	Driven by MCICLKOUT.
SD_CD	input	See Figure 18
SD_WRP	input	See Figure 18
SD_IRQ	bi-dir	See Figure 18
SD_CSn	bi-dir	See Figure 18
SD_DAT	bi-dir	See Figure 18
SD_DI	bi-dir	See Figure 18
SD_DO	bi-dir	See Figure 18

Table 23: SDCard Connections

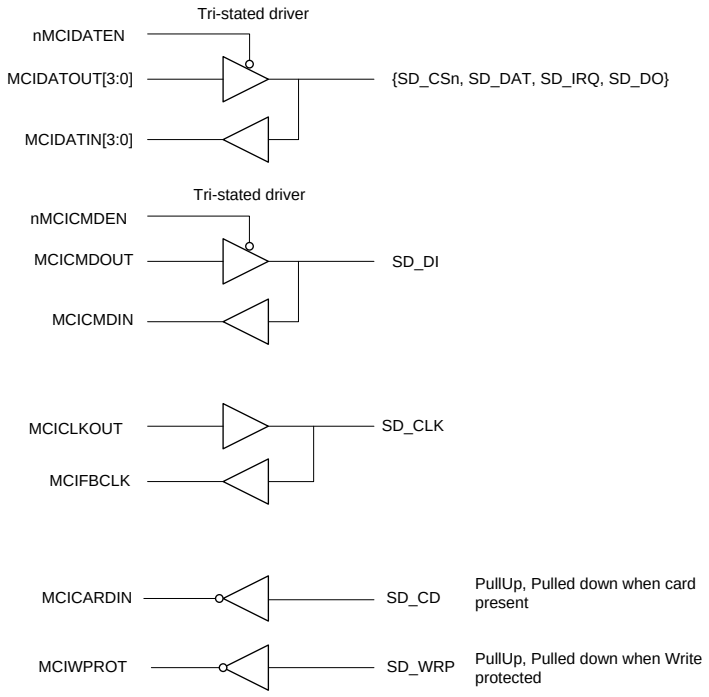


Figure 21: MCI interface connections

Copyright © 2009 ARM Limited. All rights reserved.

The nCARDIN and WPROT signals are feed to the peripheral system registers for use as setting interrupts and detecting the status of the signals.

6.9 Audio AC97 Interface

This interface is driven by the Primecell PL041 (see section 3.3.11) and connects to the AC97 Codec (National Semiconductor LM4549B) which has implemented the Line-in, line-out on the rear panel and the internal speaker.

Signal	Direction [Width]	Note
AC_BITCLK	input	12.288MHz clock from Codec
AC_EAPD	input	Not used
AC_EXT_CLK	output	24.576MHz reference clock to Codec
AC_SDATAIN	input	Drives AACISDATAIN.
AC_SDATAOUT	output	Driven by AACISDATAOUT.
AC_RESETn	output	Driven by AACIRESET.
AC_SYNC	output	Driven by AACISYNC – used as an output.

Table 24: AC97 Connections

6.10 A/D & D/A Interface

This interface is driven by the serial interface block DS702 (see section 3.3.2) and connects to the A/D and D/A converters (Cypress CY8C27543) on the rear panel.

Signal	Direction [Width]	Note
ADDA_CLK	OD	Driven by SCL.
ADDA_DATA	bi-dir	Drives SDAin and is driven to ‘0’ by nSDAOUTEN going to ‘0’

Table 25: A/D & D/A Connections

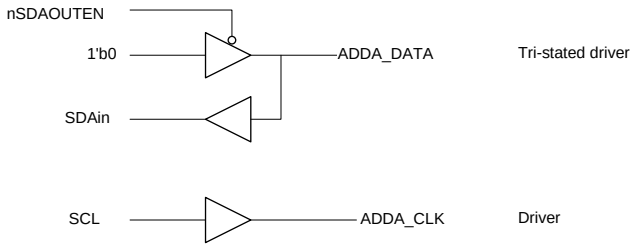


Figure 22: I²C Connections

6.11 Memory DDR Interface

The signals are connected to the Childboard connector. They reference standard DDR2 signals, but can be driven with different signals depending on the Childboard connected. This interface is not driven in the example design supplied.

Signal	Direction [Width]	Note
MEM_DDR2_ADDR	output[15:0]	
MEM_DDR2_BA	output[2:0]	
MEM_DDR2_CASn	output	

Copyright © 2009 ARM Limited. All rights reserved.

MEM_DDR2_RASn	output	
MEM_DDR2_WEn	output	
MEM_DDR2_CSn	output[1:0]	
MEM_DDR2_ODT	output[1:0]	
MEM_DDR2_CKE	output[1:0]	
MEM_DDR2_CLKP	output[1:0]	
MEM_DDR2_CLKN	output[1:0]	
MEM_DDR2_DM	output[3:0]	
MEM_DDR2_DQSN	bi-dir[3:0]	
MEM_DDR2_DQSP	bi-dir[3:0]	
MEM_DDR2_DQ	bi-dir[31:0]	
MEM_VAR	bi-dir[23:0]	

Table 26: Memory/Childboard Connections

6.12 Ethernet Phy Interface

This interface is not driven in the example design supplied, but goes directly to an Ethernet Phy (Intel LXT971A) for the user to add their own design. It implements an MII interface between the FPGA and Ethernet phy.

Signal	Direction [Width]	Note
ETH_COL	input	
ETH_CRS	input	
ETH_MDC	output	
ETH_MDIO	bi-dir	
ETH_MDINTRn	OD	
ETH_RESETn	bi-dir	
ETH_RXCLK	input	
ETH_RXD	input[3:0]	
ETH_RXDV	input	
ETH_RXER	input	
ETH_TXCLK	input	
ETH_TXD	output[3:0]	
ETH_TXEN	output	
ETH_TXER	output	

Table 27: Ethernet Phy Connections

6.13 CAN Interface

This interface is not implemented in the example design supplied, but is connected to a PHY (NXP TJA1040) for the user to add their own design.

Signal	Direction [Width]	Note
CAN_RXD	input	
CAN_TXD	output	
CAN_STB	input	

Table 28: CAN Connections

6.14 Flexray Interface

This interface is not implemented in the example design supplied, but is connected to a PHY (NXP TJA1080) for the user to add their own design.

Signal	Direction [Width]	Note
FLEX_BGE	output	
FLEX_EN	output	
FLEX_ERRn	input	
FLEX_RXD	input	

Copyright © 2009 ARM Limited. All rights reserved.

FLEX_RXEN	Input	
FLEX_STBn	output	
FLEX_TXD	output	
FLEX_TXEN	output	
FLEX_WAKE	output	

Table 29: Flexray Connections

6.15 LIN Interface

This interface is not implemented in the example design supplied, but is connected to a PHY (NXP TJA1020) for the user to add their own design.

Signal	Direction [Width]	Note
LIN_RXD	input	
LIN_TXD	output	
LIN_SLPn	output	
LIN_ACTIVE	output	

Table 30: LIN Connections

6.16 RS232 connections

These interfaces are driven by the Primecell PL011. (see section 3.3.10). There are three UARTs in the example system connected to the DUT FPGA. Interface RS0 connects to UART 1, UARTS 2 and 3 are connected to the UART ports RS232-1 and RS232-2 on the baseboard respectively (RS0_xxx_MIDI and RS1_xxx_MIDI).

Signal	Direction [Width]	Note
RS0_RXD_LVTTL	input	
RS0_TXD_LVTTL	output	
RS0_CTS_LVTTL	input	
RS0_RTS_LVTTL	output	
RS0_RXD_MIDI	input	
RS0_TXD_MIDI	output	
RS1_RXD_MIDI	input	
RS1_TXD_MIDI	output	

Table 31: RS232 Connections

Copyright © 2009 ARM Limited. All rights reserved.